В.Н. ЛЕБЕДЕВ
А.П. СОКОЛОВ

ВВЕДЕНИЕ В СИСТЕМУ ПРОГРАММИРОВАНИЯ ОС ЕС

В. Н. ЛЕБЕДЕВ,
А. П. СОКОЛОВ

ВВЕДЕНИЕ В СИСТЕМУ ПРОГРАММИРОВАНИЯ ОС ЕС

МОСКВА «СТАТИСТИКА» 1978

Л33

Лебедев В. Н. и Соколов А. П.

Введение в систему программирования ОС ЕС. М., «Статистика», 1978.

144 с. с ил.

В книге даны основы функционирования системы программирования ОС ЕС, описаны язык управления заданиями, наборы данных, опций и управляющие предложения системных обрабатывающих программ в объеме, достаточном для использования системы программирования в типовых практических случаях трансляции, редактирования, загрузки и исполнения прикладных программ.

Книга предназначена для пользователей ОС ЕС, знакомых с программированием на одном из входных языков данной системы. Она может быть использована для самостоятельного изучения, а также служить учебным пособием на курсах программирования для ЕС ЭВМ.

Л $\frac{30502\text{-}017}{008(01)\text{-}78}$ 77-78

6Ф7.3

© Издательство «Статистика», 1978

ПРЕДИСЛОВИЕ

Книга предназначена для пользователей ОС ЕС, начинающих освоение системы. Авторы не стремились исчерпывающе изложить детали ее построения и эксплуатации. Во-первых, это невозможно сделать в рамках одной книги, поскольку система программирования ОС ЕС чрезвычайно сложна, а ее подробное описание занимает тысячи страниц технической документации. Во-вторых, как показывает опыт, освоить сразу все возможности системы программирования ОС ЕС начинающий пользователь просто не в состоянии, в то же время для начала практической работы с системой достаточно ознакомиться со сравнительно небольшим кругом основных понятий и практических приемов.

Своей главной задачей авторы считали выделение важнейших понятий и изложение их в объеме, позволяющем практически использовать систему программирования ОС ЕС в типовых случаях. При этом имелось в виду, что читатель знаком по крайней мере с одним из входных языков программирования ОС ЕС и имеет общее представление о средствах всех входных языков системы программирования ОС ЕС. Дополнительные сведения о последней имеются в книгах по ЕС ЭВМ, указанных в списке использованной литературы. Кроме того, читателю может быть полезна информация о системе IBM/360. Для углубленного изучения системы программирования следует обратиться к эксплуатационной документации ОС ЕС, сочетая изучение документации с практической работой на машине.

Глава 1

СИСТЕМЫ ПРОГРАММИРОВАНИЯ ЕС ЭВМ

1.1. ОБЩИЕ СВЕДЕНИЯ

1.1.1. Классификация систем программирования

Система программирования представляет собой совокупность средств разработки программ (языки программирования, трансляторы, библиотеки подпрограмм и обслуживающие (сервисные) программы), обеспечивающих автоматизацию составления и отладки программ пользователей.

Системы программирования классифицируют по разным признакам. В частности, по набору входных языков различают *одноязыковые* и *многоязыковые* системы. Отличительная черта последних состоит в том, что отдельные части программы можно написать на разных языках, а затем с помощью системных средств (трансляторы, редактирующие программы) объединить в готовую к применению программу. Если употребляется несколько входных языков, а объединять программы, написанные на разных языках, невозможно, то это значит, что на машине имеется несколько одноязыковых систем программирования.

По возможностям расширения многоязыковые системы делят на *открытые* и *замкнутые*. К открытой системе программирования можно добавить новый входной язык с его транслятором, не внося в систему никаких внутренних изменений. Добавление нового транслятора к замкнутой системе невозможно без изменения внутреннего устройства системы. Программные компоненты открытой системы чаще всего не входят в управляющую программу (являются внешними по отношению к ней) и выполняются под контролем управляющей программы на тех же основаниях, что любые программы пользователей. Программы замкнутой системы программирования, как правило, встроены в управляющую программу операционной системы и выполняются на правах ее подпрограмм.

В операционных системах ЕС ЭВМ применяются многоязыковые открытые системы программирования.

На большинстве моделей ЕС ЭВМ, в частности на всех моделях, выпускаемых в СССР, используют одну из двух операционных систем: дисковую операционную систему ДОС ЕС и операционную систему ОС ЕС. Этим двум операционным системам соответствуют две системы программирования: система программирования ДОС ЕС и система программирования ОС ЕС. По входным языкам они частично совместимы. Хотя набор средств некоторых

входных языков программирования ДОС и ОС ЕС различен, за счет введения определенных ограничений во входные языки можно обеспечить возможность обработки одних и тех же программ на этих входных языках обеими системами. По выходным (объектным) программам ДОС и ОС ЕС несовместимы. Это означает, что программы, подготовленные системой программирования ДОС, не могут выполняться под управлением ОС, и наоборот. Несовместимость объектных программ связана с различиями в составе и функциях компонент управляющих программ ДОС и ОС. Вследствие этих различий в ДОС и ОС ЕС используются разные системные подпрограммы, обеспечивающие связь программы пользователя с операционной системой.

ДОС ЕС позволяет эффективно использовать только младшие модели ЕС ЭВМ. ОС ЕС можно применять на всех отечественных моделях ЕС ЭВМ, имеющих достаточную память. Операционная система ОС ЕС существенно превосходит ДОС ЕС по набору возможностей. Однако в полной мере они проявляются на машинах со средней и большой основной памятью и достаточной внешней памятью. В связи с увеличением памяти новых моделей ЕС ЭВМ следует ожидать, что в ближайшие годы на большинстве моделей ЕС ЭВМ будет использоваться ОС ЕС.

1.1.2. Основные понятия

Все трансляторы в системах программирования ЕС ЭВМ работают по принципу компиляции (трансляция отделена от исполнения), поэтому трансляторы для языков высокого уровня называют *компиляторами*. Транслятор для языка Ассемблера называют *Ассемблером*.

Системы программирования ЕС ЭВМ основаны на модульном принципе, поэтому программист может разделить программу на функционально независимые части — модули. Модуль, записанный на исходном языке программирования, называют *исходным модулем*. Программа может состоять из одного или нескольких исходных модулей.

Каждый исходный модуль транслируется независимо. Транслятор перерабатывает его в *объектный модуль*. Исходные модули, входящие в состав одной программы, могут быть написаны на разных языках программирования, однако на практике такая возможность несколько ограничена (см. разд. 5.3).

Объектные модули, вырабатываемые разными трансляторами, однотипны с точки зрения дальнейшей обработки, причем структура объектных модулей в ДОС и ОС ЕС одинакова. Объектный модуль содержит текст программы на машинном языке и дополнительную информацию, обеспечивающую объединение этого модуля с другими независимо транслированными модулями и настройку модуля по месту его загрузки. Непосредственно объектный модуль выполнить на машине нельзя: предварительно он должен быть отредактирован.

Редактирование объектных модулей в ДОС и ОС ЕС выполняет *редактор связей*, но его функции в этих системах различны. Это связано с различием способов выделения памяти для программ пользователей в ДОС и ОС ЕС.

В ДОС ЕС программы выполняются в двух режимах: однопрограммном и мультипрограммном. В однопрограммном режиме выполняемой программе предоставляется вся основная память, кроме той ее части, которая занята управляющей программой ДОС. В мультипрограммном режиме одновременно можно выполнять до трех программ. Каждой программе выделяется раздел основной памяти. Разделы фиксированы и назначены заранее (при генерации системы или при начальной загрузке управляющей программы). В обоих режимах перед редактированием объектного модуля известен адрес основной памяти, начиная с которого будет загружена для выполнения редактируемая программа. Поэтому редактор связей ДОС ЕС выполняет две функции: объединяет независимо транслированные объектные модули в одну программу, разрешая межмодульные ссылки, и настраивает редактируемую программу по месту ее загрузки для выполнения.

Результат работы редактора связей ДОС ЕС — *абсолютный модуль* — представляет собой готовую программу, которая помещается в библиотеку абсолютных модулей на диске (рис. 1.1,*а*). Для выполнения этой программы ее нужно только загрузить в основную память, начиная с заранее известного адреса.

Управляющая программа ОС ЕС в зависимости от объема основной памяти предусматривает три режима выполнения программ пользователей: однопрограммный режим PCP (Primary Control Program — первичная управляющая программа) при объеме основной памяти не менее 64К (К — килобайт — 1024 байт); мультипрограммный режим с фиксированным числом задач MFT (Multiprogramming Fixed Task) при объеме основной памяти не менее 128К; мультипрограммный режим с переменным числом задач MVT (Multiprogramming Variable Task) при объеме основной памяти не менее 256К.

Указанные здесь минимальные размеры основной памяти обеспечивают только техническую возможность использования соответствующего режима. Практически для эффективного применения каждого режима эти размеры нужно увеличить по крайней мере вдвое.

Основной режим ОС ЕС — мультипрограммный, поскольку он обеспечивает наиболее эффективное использование ресурсов вычислительной системы.

Одна из особенностей мультипрограммных режимов ОС ЕС состоит в том, что управляющая программа ОС выделяет память для исполнения программы пользователя непосредственно перед ее загрузкой в основную память. Вследствие этого в отличие от ДОС ЕС редактор связей ОС ЕС не может заблаговременно настроить редактируемую программу по месту загрузки. По этой причине вместо абсолютного модуля, полностью готового для вы-

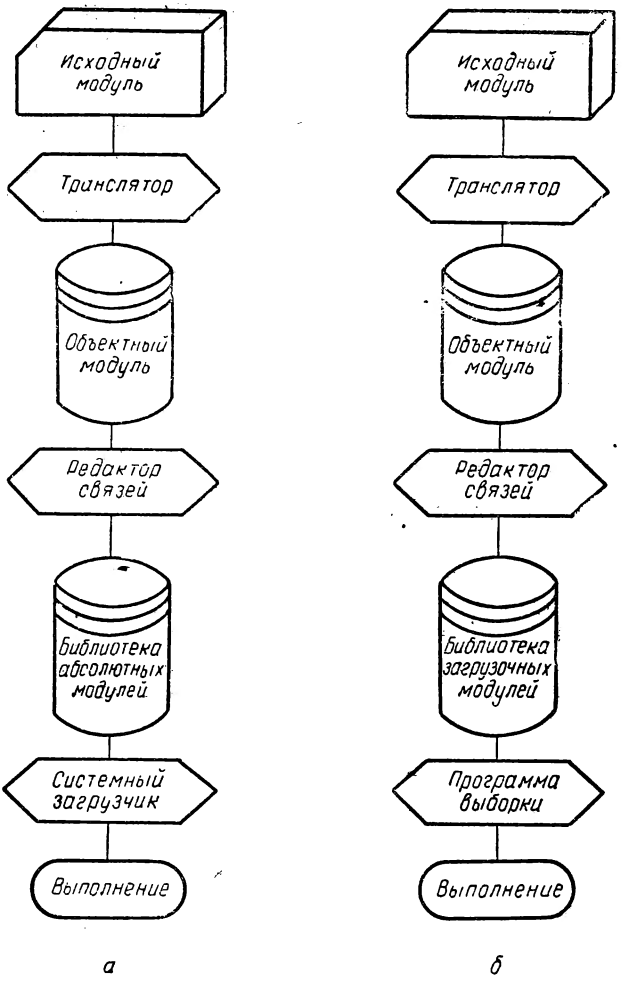


Рис. 1.1. Схема обработки исходного модуля:
а) в ДОС ЕС, *б)* в ОС ЕС.

полнения, редактор связей ОС ЕС вырабатывает *загрузочный модуль,* структурно подобный объектному модулю.

Обычно загрузочный модуль представляет собой объединение объектных модулей или объектных и ранее отредактированных загрузочных модулей. В загрузочном модуле, как правило, разрешены все межмодульные ссылки, но он, как и объектный модуль, не настроен по месту загрузки. Чтобы получить из загрузочного модуля готовую программу, нужно настроить по месту загрузки константы, значения которых зависят от адреса загрузки *(адресные константы).*

Загрузку в основную память загрузочного модуля и корректировку адресных констант выполняет программа выборки, входящая в состав управляющей программы. Загрузив программу пользователя в память, программа выборки передает ей управление.

Трансляцией, редактированием и выполнением программы пользователя управляет операционная система. Для указания ей требуемых действий нужно составить *задание*. В задании указывают программы, которые нужно выполнить, режимы выполнения программ, а также данные и устройства, с которыми работает каждая программа. В частности, согласно схеме, представленной на рис. 1.1,*б*, нужно последовательно выполнить программу транслятора, программу редактора связей и программу пользователя, которая загружается программой выборки.

Задание операционной системе записывают на языке управления заданиями. Предложения этого языка называют *управляющими операторами*. Последовательность перфокарт с управляющими операторами и перфокарт с данными, поступающих в систему через специально выделенное устройство системного ввода, образует *входной поток заданий*. Задания интерпретируются программой управления заданиями, составляющей часть управляющей программы операционной системы.

Программы, выполняющие обработку данных под управлением операционной системы, в ОС ЕС называют *обрабатывающими*. Обрабатывающие программы, входящие в состав ОС ЕС, в частности трансляторы и редактор связей, называют *системными*, а все прочие обрабатывающие программы *прикладными*, или *проблемными*.

1.2. СИСТЕМА ПРОГРАММИРОВАНИЯ ОС ЕС

1.2.1. Наборы данных

Функционирование системы программирования ОС ЕС опирается на наборы данных, которыми управляет операционная система.

Набором данных в ОС ЕС называют поименованную совокупность данных, объединенных общим назначением, имеющую определенную организацию. Наборы данных располагаются на внешних носителях.

В ОС ЕС наборы данных могут быть временными и постоянными. *Временные наборы* создаются при выполнении отдельных заданий. Они существуют до завершения задания, в котором они созданы. *Постоянные наборы* данных создаются либо при генерации системы, либо в отдельных заданиях. Наборы данных, созданные при генерации системы, хранятся всегда. Постоянный набор данных, созданный в задании, сохраняется после его завершения до тех пор, пока системе не будет дано указание уничтожить его.

Постоянные наборы данных делят на *каталогизированные* и *сохраняемые* (некаталогизированные). Сведения о местоположе-

нии каталогизированного набора данных хранятся в каталоге системы. Для доступа к такому набору данных достаточно указать его имя. Сведения о местоположении сохраняемых наборов данных в системе не хранятся. Для доступа к этому набору данных нужно указать не только его имя, но и местоположение (устройство, номер диска или бобины магнитной ленты).

По способу организации различают последовательные, библиотечные, индексно-последовательные наборы данных и наборы данных с прямой организацией. Программы системы программирования работают только с последовательными и библиотечными наборами данных, поэтому индексно-последовательные наборы данных и наборы данных с прямой организацией здесь не рассматриваются.

Последовательный набор данных SDS (Sequential Data Set) состоит из отдельных записей или блоков записей, расположенных на носителе последовательно. Объединение записей в блоки (блокирование) применяют для экономии места на носителе (блокирование сокращает количество неиспользуемых промежутков между записями). Чтобы отыскать конкретную запись, нужно считывать записи (блоки записей) подряд, пока не будет найдена нужная.

Последовательный набор данных может располагаться на любом носителе: перфокартах, перфоленте, магнитной ленте, диске или барабане. Для обращения к последовательному набору данных нужно указать его имя.

Библиотечный набор данных PDS (Partitioned Data Set), или *библиотека,* состоит из именованных *разделов*. Разделы называют иногда членами библиотеки. Записи в разделе располагаются подряд, как в последовательном наборе данных. Имена разделов и адреса их хранения перечислены в оглавлении, размещенном в начале библиотеки. Записи можно добавлять в библиотеку или считывать из нее только целыми разделами. Для обращения к разделу библиотеки нужно указать имя библиотеки и имя раздела. Обычно в библиотеках хранятся программы. В литературе по ОС ЕС библиотеки называют также разделенными или расчлененными наборами данных. Библиотеки в ОС ЕС располагаются только на устройствах прямого доступа (обычно на дисках).

В ОС ЕС, как и в ДОС, имеются системные и личные библиотеки. Кроме того, в ОС ЕС существуют временные библиотеки.

Системные библиотеки — это постоянные каталогизированные библиотечные наборы данных, которые хранят системные программы самой ОС ЕС, а также системные подпрограммы. В общей системной библиотеке SYS1.LINKLIB находятся системные обрабатывающие программы, в том числе трансляторы и редактор связей. Здесь могут также размещаться часто используемые прикладные программы. Общая системная библиотека всегда доступна для применения в любом задании. Чтобы воспользоваться в задании какой-либо другой системной библиотекой, например

библиотекой определенного транслятора, нужно сообщить операционной системе имя этой библиотеки.

Личные библиотеки — это постоянные каталогизированные или сохраняемые библиотечные наборы данных, которые создаются пользователями и хранят их программы. Для использования в задании личной библиотеки нужно указать ее имя, а для сохраняемой библиотеки и местоположение.

Временные библиотеки — это временные библиотечные наборы данных, существующие только в задании, в котором они созданы. В частности, во временных библиотеках обычно хранят созданные в задании загрузочные модули перед их выполнением.

Последовательные и библиотечные наборы данных играют важную роль в функционировании системы программирования ОС ЕС: исходный модуль перед трансляцией помещают в последовательный набор данных, объектные модули могут находиться как в последовательных, так и в библиотечных наборах данных. Загрузочные модули перед исполнением хранятся только в библиотеках, причем каждый загрузочный модуль составляет один раздел библиотеки.

1.2.2. Состав и схема функционирования системы программирования

Систему программирования ОС ЕС образуют входные языки программирования с их трансляторами, редактор связей, загрузчик, средства отладки, а также некоторые системные библиотеки, принадлежащие трансляторам или содержащие часто используемые подпрограммы. Кроме того, к системе программирования относят отдельные вспомогательные программы (утилиты), предназначенные для обслуживания библиотек и других наборов данных.

В системе программирования ОС ЕС имеются следующие входные языки: язык Ассемблера, Фортран, Кобол, ПЛ/1, РПГ и Алгол-60. Как уже упоминалось, программа на входном языке перед выполнением должна быть последовательно обработана транслятором и редактором связей. Поэтому основными компонентами системы программирования являются трансляторы и редактор связей, а также загрузчик, объединяющий в упрощенном варианте функции редактора связи и программы выборки. Эти системные обрабатывающие программы используют определенные входные, промежуточные и выходные наборы данных (рис. 1.2).

Исходная программа, написанная на одном из входных языков, в виде исходного модуля помещается во входной набор данных соответствующего транслятора. Этот набор данных обычно является последовательным и записывается на перфокартах, но он может также находиться на магнитной ленте или на диске.

Ассемблер и почти все компиляторы в ходе трансляции используют промежуточные последовательные наборы данных на диске или на ленте.

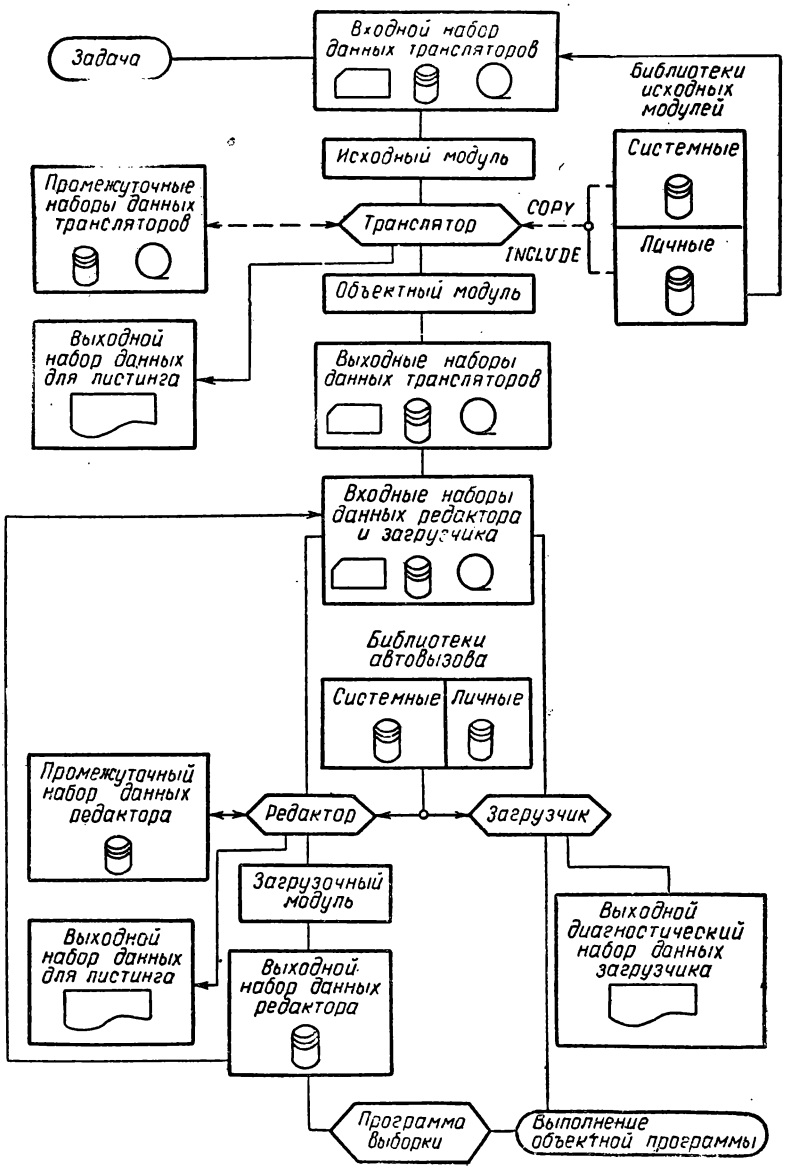


Рис. 1.2. Схема функционирования системы программирования ОС ЕС

В процессе трансляции Ассемблер, а также компиляторы Кобола и ПЛ/1 могут включать в транслируемую программу тексты на исходных языках из библиотек. Ассемблер делает это, обрабатывая оператор COPY или библиотечную макрокоманду, компилятор Кобола — обрабатывая пункт COPY и утверждение INCLUDE, а компилятор ПЛ/1 — встретив в исходной программе утверждение %INCLUDE.

Все трансляторы вырабатывают однотипный с точки зрения редактора связей объектный модуль и помещают его в выходной последовательный или библиотечный набор данных. (Структура объектного модуля описана в п. 4.1.1.) Если полученный объектный модуль нужен только в текущем задании и его не предполагается сохранять для будущего использования, то в качестве выходного набора данных транслятора назначают временный последовательный набор данных или временную библиотеку. Вид выходного набора данных транслятора программист указывает в задании.

В ходе трансляции по указанию программиста Ассемблер и компиляторы могут формировать в отдельном выходном последовательном наборе данных листинг и диагностические сообщения для последующей выдачи на печать.

Выработанный транслятором объектный модуль может быть передан на редактирование или на загрузку и выполнение. Для этого объектный модуль или несколько объектных модулей, подлежащих объединению, нужно поместить во входной набор данных редактора связей (загрузчика). Входной набор данных редактора связей (загрузчика) может размещаться на перфокартах, магнитном диске или магнитной ленте. Если объектные модули находятся на магнитном носителе (например, после работы транслятора), то физически перемещать их не нужно. Достаточно объявить в задании имена наборов данных (постоянных или временных), образующих входной набор данных редактора связей (загрузчика).

В загрузочный модуль редактор связей может включить не только объектные модули, но и ранее отредактированные загрузочные модули. Если в конце редактирования в загрузочном модуле остались неразрешенные внешние ссылки к другим модулям, то редактор связей может организовать поиск недостающих объектных или загрузочных модулей в системных и личных библиотеках, имена которых программист указал в задании редактору связей. Эти библиотеки называют *библиотеками автовызова.* По своему усмотрению программист может запретить использование библиотек автовызова.

В ходе редактирования используется промежуточный последовательный набор данных на диске. Сформированный загрузочный модуль редактор связей помещает в постоянный или временный библиотечный выходной набор данных на диске. Этот набор данных должен быть описан в задании. Дополнительно по указанию программиста редактор связей может формировать в отдельном

выходном последовательном наборе данных листинг, выдаваемый на печать.

После редактирования исполнение полученного загрузочного модуля организует программа выборки, входящая в состав супервизора. Если задание предписывает выполнить загрузочный модуль, то программа выборки отыскивает указанный ей загрузочный модуль в библиотеке, загружает модуль в основную память, настраивает его по месту загрузки, корректируя адресные константы, и передает управление в точку входа.

Программа выборки организует поиск и выполнение всех обрабатывающих программ как прикладных, так и системных, в том числе трансляторов и редактора связей.

Если после редактирования предусматривается немедленное выполнение загрузочного модуля, то можно избежать промежуточной записи модуля на диск. Для этого вместо редактора связей нужно вызвать загрузчик, который совмещает основные функции редактора и программы выборки. Загрузчик помещает в основную память объектные и загрузочные модули, находящиеся в его входном наборе данных, объединяет их, разрешая межмодульные ссылки, корректирует адресные константы для настройки программы по месту и передает управление в точку входа сформированной программы. Эта программа не может быть помещена в библиотеку, она исполняется немедленно.

В ходе редактирования загрузчик может формировать выходной набор данных, содержащий диагностические и справочные сообщения, а также может организовать поиск и вызов недостающих объектных или загрузочных модулей из библиотек автовызова, указанных программистом. Эти действия выполняются, если они предусмотрены заданием.

Глава 2

ЯЗЫК УПРАВЛЕНИЯ ЗАДАНИЯМИ

2.1. ЗАДАНИЕ

2.1.1. Формат управляющих операторов

Для указания операционной системе последовательности программ, которые нужно выполнить, и для описания используемых этими программами наборов данных составляют задание.

Задание состоит из управляющих операторов. Обычно их записывают на перфокартах. Каждый управляющий оператор начинается с новой перфокарты. Перфокарты, содержащие управляющие операторы, называют *управляющими картами.* Управляющие карты помещают во входной поток заданий. Их вводит программа системного ввода.

Большинство управляющих операторов записывают в следующем формате:

Перед нанесением на перфокарту управляющие операторы записывают на стандартном бланке, строка которого соответствует 80-колонной перфокарте. Номера позиций бланка, отсчитываемые слева направо, совпадают с номерами колонок перфокарты, поэтому при описании формата будем говорить о номерах колонок перфокарт.

Знаки «//» должны находиться в первых двух колонках перфокарты. Они указывают, что это управляющая карта. Следующие за знаками «//» четыре поля: имени, операции, операндов и комментария — разделяют одним или несколькими пробелами. В изображении формата пробелы показаны штриховкой. Пробелы служат для указания границ полей, поэтому внутри последних, кроме поля комментария, пробелы не допускаются. Отдельные поля могут быть пустыми.

Поле имени всегда начинается в третьей колонке перфокарты. Оно содержит имя (метку) оператора. Имя есть последовательность латинских букв, цифр и знаков #, @ , ¤, начинающаяся с буквы. Русские буквы, отличные от латинских, и другие знаки для записи имени использовать нельзя. Длина имени — не более восьми знаков. Имя предназначено для ссылок на именованный оператор.

Остальные поля могут начинаться в любой колонке.

Поле операндов содержит список параметров, разделенных запятыми. Различают позиционные и ключевые параметры. Позиционные параметры записывают в начале поля операндов, а ключевые — вслед за ними. Некоторые параметры можно опускать. В этом случае опущенному параметру присваивается принятое в системе стандартное значение, которое может быть и пустым.

Каждый *позиционный параметр* записывают на отведенной для него фиксированной позиции. Если позиционный параметр опущен, то нужно записать запятую, отделяющую этот параметр от следующего позиционного параметра. Например, пусть A, B, C — позиционные параметры. Если опущен параметр A, то пишут

,B,C

Если опущен параметр B:

A,,C

Запятые вместо позиционных параметров не проставляют, если опущенные параметры последние в списке позиционных параметров или если опущены все позиционные параметры.

Ключевой параметр состоит из ключевого слова, за которым следует знак равенства и значение ключевого параметра, например в записи

SYSOUT=B

слово SYSOUT — ключевое слово, B — значение ключевого параметра.

Ключевые параметры можно записывать в произвольном порядке, причем любой ключевой параметр можно опустить, если программиста удовлетворяет его стандартное значение.

Читатель, знакомый с правилами записи макрокоманд Ассемблера, несомненно, уже заметил, что правила записи операторов в языке управления заданиями и в макроязыке Ассемблера очень похожи.

Значения некоторых позиционных и ключевых параметров задают в виде списка *подпараметров*. Как и параметры, подпараметры могут быть позиционными и ключевыми. Позиционные подпараметры записывают на определенных позициях, а порядок следования ключевых подпараметров произволен. Подпараметры в списке разделяют запятыми, а весь список заключают в апострофы или в круглые скобки. Особенности употребления апострофов и скобок будут отмечены ниже. Скобки или апострофы могут не потребоваться, если в списке записан только один подпараметр (остальные опущены).

Пример 1. Запись списка подпараметров.

PARM='MAP,LET,LIST',DISP=(OLD,PASS)

Пример 2. Список, состоящий из одного подпараметра

DISP=OLD

Обычно для записи значений параметров и подпараметров используют цифры и латинские буквы. Однако иногда значение па-

раметра (подпараметра) может содержать другие знаки, например русские буквы, знаки арифметических операций и т. п. В этих случаях значение заключают в апострофы. Имеются исключения из этого правила. Все они специально оговорены в описании языка управления заданиями. Если хотя бы один подпараметр заключен в апострофы, а список включает несколько подпараметров, то весь список подпараметров заключают в круглые скобки.

Пример 3. Употребление апострофов и скобок.

```
PARM=(C2,'ОШИБКА','A+B/D',9456)
```

В *поле комментария* можно поместить произвольный текст. Обычно это пояснения, которые программист считает нужным привести в операторе задания. Текст комментария выдается в листинге задания, если предусмотрена печать управляющих операторов. На выполнение оператора текст не влияет.

Все управляющие операторы, кроме специального оператора комментария, о котором будет сказано в следующем пункте, записывают на перфокарте не далее колонки 71. При необходимости поле операндов и поле комментария можно продолжить на нескольких картах. *Карта продолжения* должна иметь знаки «//» в первой и второй колонках. Прерванное поле операндов обязательно заканчивается запятой, находящейся после параметра или подпараметра. Список подпараметров можно прерывать лишь в том случае, когда он заключен в скобки. Список подпараметров, заключенный в апострофы, разрывать нельзя.

Поле операндов продолжают начиная с колонок 4—16. Пробел после параметра (а не запятая) означает окончание поля операндов. На карте с прерванным полем операндов в колонке 72 можно поставить *знак продолжения* — любой знак, отличный от пробела, но это не обязательно.

Поле комментария можно прервать в любом месте, поставив в колонке 72 знак продолжения, здесь он обязателен. На карте продолжения поле комментария можно продолжить с любой колонки после колонки 3.

2.1.2. Структура задания

Язык управления заданиями ОС ЕС включает девять управляющих операторов (табл. 2.1).

Основные операторы языка управления заданиями: оператор задания (оператор JOB*), оператор пункта задания (оператор EXEC) и оператор описания данных (оператор DD). Эти три оператора используются практически в любом задании, остальные имеют вспомогательное значение. Чаще других применяют оператор ограничения и пустой оператор.

---

* Названия основных операторов происходят от английских слов: JOB — задание, EXEC — сокращение слова execute — выполнять, DD — первые буквы слов data definition — определение данных.

Таблица 2.1

Управляющие операторы языка управления заданиями ОС ЕС

Название оператора	Формат оператора
Оператор задания	//имя JOB [операнды] [комментарий]
Оператор пункта задания	//[имя] EXEC операнды [комментарий]
Оператор описания данных	//[имя] DD операнды [комментарий]
Оператор ограничения	/* [комментарий]
Пустой оператор	//
Оператор комментария	//*комментарий
Оператор процедуры	//[имя] PROC операнды [комментарий] (каталогизированная процедура) //имя PROC [операнды] [комментарий] (процедура во входном потоке)
Оператор конца процедуры	//[имя] PEND [комментарий]
Оператор команды	//команда операнды [комментарий]

Примечание. В квадратные скобки заключены названия необязательных полей, которые можно опустить.

Оператор команды предназначен для ввода через входной поток команд управления системой, которые обычно передаются через пультовую пишущую машинку. На практике программисты этот оператор не используют.

Оператор комментария включает в задание пояснительный текст, который может выдаваться на печать в листинге. На выполнение задания этот оператор не влияет. Знаки «//*» помещаются в первых трех колонках перфокарты, а текст комментария — в колонках 4—80. Оператор комментария всегда занимает одну перфокарту.

Оператор процедуры и оператор конца процедуры служат для описания процедур. *Процедура* — это последовательность управляющих операторов, которая находится во входном потоке или в специальной библиотеке процедур SYS1.PROCLIB. Процедуру, хранящуюся в библиотеке процедур, называют *каталогизированной.*

Оператор JOB всегда является первым оператором задания. Задание состоит из одного или нескольких пунктов. Каждый пункт начинается оператором EXEC. Он указывает программу или процедуру, которую нужно выполнить в пункте задания. В числе операторов процедуры непременно есть хотя бы один оператор EXEC, определяющий программу, поэтому в конечном итоге этот оператор всегда указывает (прямо или через процедуру) выполняемую программу.

После оператора EXEC записывают операторы DD, которые определяют наборы данных, используемые в пункте задания. Иногда операторы DD могут располагаться также вслед за оператором JOB. Такие операторы DD описывают специфичные наборы данных (например, библиотеки), используемые всеми пунктами задания.

Конец задания и начало следующего задания указывает очередной оператор JOB. Аналогично, конец пункта задания указывает следующий оператор EXEC или очередной оператор JOB, начинающий новое задание. Следовательно, в каждом задании есть только один оператор JOB, а в каждом пункте задания — только один оператор EXEC.

Если задание последнее в потоке заданий (других заданий нет), то его конец отмечают пустым оператором //. Другого назначения пустой оператор не имеет.

Во входном потоке наряду с управляющими картами могут также находиться карты с исходными данными программ, выполняемых в задании. Исходные данные помещают в том пункте задания, к которому они относятся. Картам с исходными данными всегда предшествует оператор DD, имеющий специальный позиционный параметр в поле операндов, а конец исходных данных отмечают оператором ограничения /*. Другого назначения оператор ограничения не имеет. Структура задания, состоящего из двух пунктов, показана на рис. 2.1.

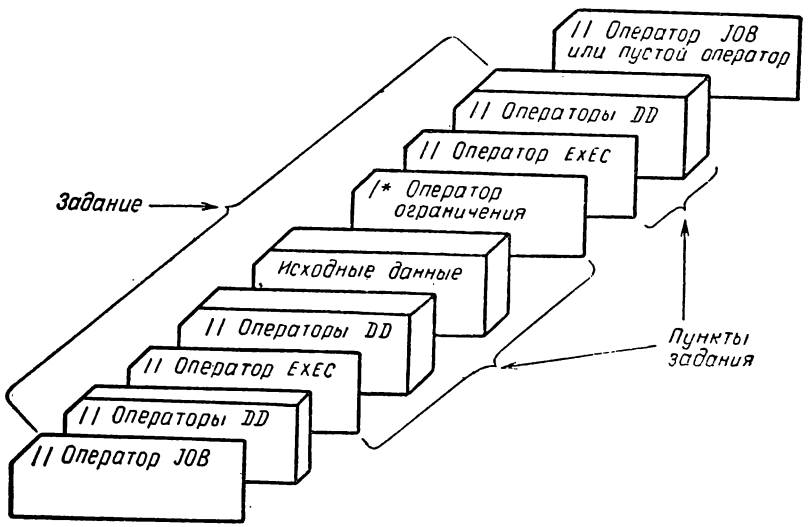


Рис. 2.1. Структура задания в ОС ЕС

Синтаксис управляющих операторов прост, однако большое число параметров и подпараметров, используемых в поле операндов, затрудняет изучение языка управления заданиями. Вместе с тем многие параметры и подпараметры имеют в системе стандартные значения, поэтому в типовых случаях их можно явно не указывать. По этой причине при описании управляющих операторов рассматриваются только наиболее важные параметры. В частности, в этой главе приведены все параметры и подпараметры, используемые в каталогизированных процедурах трансляции, редак-

тирования и выполнения, которыми часто пользуются программисты. Перечень и краткое описание всех параметров управляющих операторов задания, пункта задания и описания данных приведены в приложении 1.

2.2. ОПИСАНИЕ ЗАДАНИЯ

2.2.1. Оператор задания

Оператор JOB должен обязательно содержать в поле имени имя задания, причем в мультипрограммных режимах имена заданий, одновременно находящихся в системе, должны быть различны. В поле операндов можно записать два позиционных параметра и одиннадцать ключевых. В общем случае все эти параметры не обязательны.

Первый позиционный параметр — учетная информация, второй — имя программиста. Учетная информация и имя программиста обязательны лишь тогда, когда в операционную систему включена специальная программа обработки учетной информации. В этом случае администрация вычислительного центра устанавливает определенные правила записи позиционных параметров. Задания, в которых эти правила не соблюдаются, операционная система не выполняет.

Учетная информация есть список подпараметров, разделенных запятыми. Список заключают в апострофы или круглые скобки. Первый подпараметр — учетный номер, остальные подпараметры — дополнительная учетная информация. Если имеется только учетный номер и в нем нет специальных знаков (например, нет русских букв), то его можно не заключать в апострофы или скобки. Учетная информация может содержать до 142 знаков, включая запятые.

Имя программиста — это либо фамилия программиста, либо код задачи. Если имя программиста содержит специальные знаки (например, русские буквы), то его нужно заключить в апострофы. Заметим, что в имени программиста точки не считаются специальными знаками. Число знаков в имени программиста не должно превосходить 20.

Пример 1. Запись учетной информации и имени программиста.

```
//EXAC1 JOB(501,7—70),'ЛЕБЕДЕВ В.Н.'
```

Пример 2. Запись имени программиста без учетной информации.

```
//EXAC2 JOB ,'СОКОЛОВ А.П.'
```

Запятая перед именем программиста обязательна, потому что первый позиционный параметр опущен.

Из ключевых параметров рассмотрим только четыре: MSGLEVEL, TIME, REGION и COND.

Параметр MSGLEVEL (message level — уровень сообщений) определяет уровень вывода сообщений системы в процессе выполнения задания. Формат параметра:

MSGLEVEL=(операторы, сообщения)

Подпараметр «операторы» имеет три значения: 0, 1 и 2. Если вместо «операторы» записать 0, то печатается только оператор JOB. Значение 2 предписывает печатать все входные операторы задания, а значение 1 — все входные операторы, операторы каталогизированных процедур, а также операторы каталогизированных процедур после замены символических параметров (процедуры и их символические параметры описаны в п. 2.4.1).

Подпараметр «сообщения» имеет два значения: 0 и 1. Значение 0 означает, что не нужно печатать сообщения о распределении устройств для размещения наборов данных и о состоянии наборов данных, а значение 1 предписывает печатать эти сообщения. При аварийном завершении задания сообщения о распределении ресурсов печатаются всегда.

Обычно в системе принято стандартное значение MSGLEVEL=(0,1). Поэтому если параметр MSGLEVEL опущен, то будут печататься только операторы JOB и все сообщения о распределении. При указанном стандартном значении параметра MSGLEVEL записи MSGLEVEL=(2,1) и MSGLEVEL=2 эквивалентны, поскольку пропуск второго подпараметра означает, что ему нужно присвоить стандартное значение. По аналогичной причине эквивалентны записи MSGLEVEL=(0,0) и MSGLEVEL=(,0).

Параметр TIME в мультипрограммных режимах устанавливает верхний предел времени выполнения задания центральным процессором. Это позволяет избежать напрасных затрат машинного времени в случае зацикливания программы. Задание, превысившее отведенное время, снимается, если в системе при ее генерации не предусмотрены другие действия. Формат параметра:

TIME=(минуты,секунды)

Подпараметр «минуты» не должен превосходить 1440 (число минут в сутках), а подпараметр «секунды» должен быть меньше 60. Обычно в системе установлено стандартное значение TIME=30 (30 мин).

Запись TIME=(1,30) означает «1 минута 30 секунд». Запись TIME=1440 снимает ограничение на время выполнения задания.

Параметр REGION указывает максимальную величину области основной памяти, необходимую для выполнения всех пунктов задания. Этот параметр используется только в режиме MVT (мультипрограммирование с переменным числом задач). В остальных режимах параметр REGION игнорируется. В простейшем случае параметр REGION имеет вид:

REGION=величинаK

«Величина» определяет четное число килобайт памяти. Если указано нечетное число килобайт, то система округляет его в

большую сторону до четного. Обычно при генерации системы устанавливается стандартное значение REGION=50K.

Параметр COND (condition — условие) задает условия прекращения выполнения задания.

Любая обрабатывающая программа после завершения может записать в регистр 15 *код возврата* — число, имеющее значение от 0 до 4095 и кратное 4. В прикладных обрабатывающих программах действия, формирующие код возврата и его запись в регистр 15, предусматривает сам программист. Программист устанавливает и смысл кода возврата.

Системные обрабатывающие программы (в частности, трансляторы, редактор связей и загрузчик) вырабатывают следующие коды возврата:

0 — если в ходе выполнения программы (например, программы транслятора) ошибок не обнаружено;

4 — если обнаружена ситуация, которая может привести к ошибкам при дальнейшей обработке программы;

8 — если обнаружены ошибки, которые могут сделать дальнейшую обработку программы невозможной;

12 — если обнаружены серьезные ошибки, которые делают дальнейшую обработку программы невозможной;

16 — если обнаружены ошибки, которые привели к прекращению выполнения самой обрабатывающей программы.

После завершения каждого пункта задания управляющая программа анализирует код возврата и в соответствии с условиями, заданными в параметре COND, может прекратить дальнейшее выполнение задания. Это позволяет избежать непроизводительных затрат машинного времени.

Параметр COND имеет формат:

COND=((код,оператор),...,(код,оператор))

Подпараметр (код, оператор) задает условие прекращения выполнения задания. Допускается до восьми таких подпараметров. «Код» — целое число от 0 до 4095, «оператор» — одна из операций отношения: GT — больше, GE — больше или равно, EQ — равно, LT — меньше, LE — меньше или равно, NE — не равно. По существу, подпараметр (код,оператор) задает отношение вида

«код» «оператор» «код возврата»

Если это отношение истинно хотя бы для одного подпараметра, то выполнение задания прекращается. Например, COND= =(12,LT) предписывает прекратить выполнение задания, если код возврата превосходит 12. В этом примере внешние скобки опущены, потому что использован только один подпараметр параметра COND.

При отсутствии в операторе задания параметра COND управляющая программа не анализирует коды возврата на уровне задания.

Пример 3. Запись оператора задания без параметров.

```
//PROBLEM2 JOB
```

Пример 4. Запись оператора задания с параметрами.

```
//PROBLEM3 JOB MSGLEVEL=1,TIME=10,
//            COND=(8,LE),REGION=58K
```

Если в системе приняты описанные выше стандартные значения параметров, то в примере 3 по умолчанию принимаются такие значения: MSGLEVEL=(0,1), TIME=30, REGION=50K, а коды возврата на уровне задания не анализируются. В примере 4 оператор задания записан на двух управляющих картах. На карте продолжения параметр COND должен начинаться не ближе колонки 4 и не дальше колонки 16.

2.2.2. Оператор пункта задания

Основное назначение оператора EXEC — указать программу или процедуру, которую нужно выполнить.

В поле имени оператор EXEC может содержать имя пункта задания. Хотя это имя не обязательно, но рекомендуется именовать в задании все пункты, давая им уникальные имена, так как имена пунктов задания используются в системных сообщениях о ходе выполнения задания. Отсутствующее имя пункта заменяется в сообщении пробелами, поэтому трудно бывает определить, к какому пункту относится сообщение. Кроме того, имя пункта задания можно употреблять в других пунктах того же задания для ссылок.

В поле операндов оператора EXEC можно записать один позиционный параметр и восемь ключевых Обязательным является только позиционный параметр, так как именно он указывает программу или процедуру, которую нужно выполнить.

Позиционный параметр имеет одну из следующих четырех форм:

```
PGM=имя программы
PGM=ссылка
PROC=имя процедуры
имя процедуры
```

Параметр PGM, как и параметр PROC, по форме записи напоминает ключевой (содержит знак равенства), но на самом деле это позиционный параметр. В поле операндов он всегда должен быть записан первым.

В форме «PGM=имя программы» позиционный параметр указывает, что нужно выполнить программу, которая находится в постоянной (системной или личной) библиотеке. Если программа находится в общей системной библиотеке SYS1.LINKLIB, то никаких дополнительных указаний системе не требуется. (Способы использования программ из других постоянных библиотек подробнее рассматриваются в разд. 2.3.)

В форме «PGM=ссылка» ссылка указывает на оператор описания данных, в котором описан библиотечный набор данных, содержащий нужную программу (см. п. 2.3.2). Обычно эту форму используют для выполнения программы из временной библиотеки, созданной в том же задании.

Формы «PROC=имя процедуры» и «имя процедуры» эквивалентны. Отсутствие в начале поля операндов ключевого слова PGM указывает, что нужно выполнить процедуру. Слово PROC не обязательно. Оно только подчеркивает требование выполнить процедуру. (Использование процедур описано в разд. 2.4).

Пример 1. Вызов программы.

```
//LKED EXEC PGM=IEWL
```

Пример 2. Вызов процедуры.

```
//STEP1 EXEC PROC=FORTGC
```

Пример 3. Вызов процедуры.

```
//STEP2 EXEC FORTGC
```

Два последних примера эквивалентны.

Из ключевых параметров оператора EXEC отметим четыре: TIME, REGION, COND и PARM.

Параметры TIME и REGION по назначению и форме записи аналогичны таким же параметрам оператора JOB (см. п. 2.2.1). Однако здесь эти параметры определяют время и память для пункта задания. Параметр TIME действует только в мультипрограммных режимах. Если время, установленное этим параметром, превосходит время, указанное в параметре TIME оператора JOB, то параметр TIME пункта задания игнорируется. Параметр REGION пункта задания действует только в режиме MVT. Он игнорируется, если указан параметр REGION задания.

Параметр COND пункта задания определяет условия обхода пункта задания или условия выполнения пункта задания в зависимости от того, как закончилось выполнение предыдущих пунктов (нормально или аварийно), а также в зависимости от значений выработанных предыдущими пунктами кодов возврата.

Для указания условий обхода пункта задания записывают:

```
COND=((код,оператор,имя пункта),...,(код,оператор,имя пункта))
```

«Код» и «оператор» имеют тот же смысл, что в параметре COND оператора JOB (см. п. 2.2.1). «Имя пункта» указывает один из предыдущих пунктов задания, выдавший проверяемый код возврата. Всего можно задать до восьми подпараметров. Пункт задания обходится, если выполнено хотя бы одно из заданных условий. Например, параметр COND=((8,LE,STEP1),(8,LE,STEP2)) предписывает обойти выполнение пункта задания, если код возврата, выработанный в пункте с именем STEP1 или в пункте с именем STEP2, больше 7.

Если использован только один подпараметр, то внешние скобки можно опустить. Например, COND=(12,LE,STEP3).

Аварийное окончание одного из пунктов задания обычно прекращает выполнение остальных пунктов того же задания. Если требуется продолжить выполнение задания, то нужно заменить один из подпараметров параметра COND словом EVEN (даже) или словом ONLY (только). Подпараметр EVEN предписывает

игнорировать аварийное завершение предыдущих пунктов задания, а подпараметр ONLY — выполнять данный пункт задания только в том случае, если какой-либо из предыдущих пунктов закончился аварийно. Прочие подпараметры параметра COND имеют прежний смысл, т. е. пункт обходится, если выполнено хотя бы одно из условий (код, операция, имя пункта).

Пример 4. Безусловное выполнение пункта задания.

```
//STEP3 EXEC PGM=P,COND=EVEN
```

Программа P выполняется безусловно, как бы не закончились предыдущие пункты задания.

Пример 5. Условное выполнение пункта задания.

```
//STEP4 EXEC PGM=P1,COND=(EVEN,(12,LE,STEP1))
```

Программа P1 будет выполнена даже в том случае, если предыдущие пункты закончились аварийно, но для этого код возврата в пункте с именем STEP1 должен быть меньше 12, в противном случае пункт STEP4 выполнен не будет.

Пример 6. Выполнение пункта задания при аварийном завершении предыдущих пунктов.

```
//STEP5 EXEC PGM=P2,COND=ONLY
```

Программа P2 будет выполнена только в том случае, если хотя бы один из предыдущих пунктов задания закончится аварийно.

Пример 7. Условное выполнение пункта задания при аварийном завершении предыдущих пунктов.

```
//STEP6 EXEC PGM=P3,COND=((16,LE,STEP2),ONLY)
```

Программа P3 будет выполнена только в случае, когда хотя бы один из предыдущих пунктов закончится аварийно и одновременно код возврата в пункте STEP2 будет меньше 16.

Все подпараметры параметра COND, в том числе подпараметры EVEN и ONLY, являются ключевыми и могут находиться на любой позиции списка подпараметров.

Параметр PARM служит для передачи обрабатывающей программе, выполняемой в пункте задания, управляющей информации. Управляющая информация специфична для каждой обрабатывающей программы. Ее состав устанавливает разработчик программы. Управляющая информация для системных обрабатывающих программ (трансляторов, редактора связей и загрузчика) описана в гл. 3 и 4, а также в приложении 2. Управляющую информацию записывают в виде значения параметра PARM в таком формате:

```
PARM=значение
```

«Значение» — строка, длина которой не превосходит 100 знаков. Если строка состоит из нескольких подпараметров, разделенных запятыми, то ее заключают в апострофы или круглые скобки, которые не являются частью значения (они только выделяют его). Напомним, что подпараметр, содержащий специальные знаки, отличные от цифр и латинских букв, нужно заключить в апострофы. В этом случае все значение заключается в круглые скобки. Апостроф, являющийся в подпараметре частью значения, пишут дважды.

Например, чтобы передать значение P'2 пишут PARM='P''2'. Напомним также, что значение, заключенное в круглые скобки, можно прервать после запятой, разделяющей подпараметры и, продолжить на следующей карте, а значение, заключенное в апострофы, прерывать и переносить на другую карту нельзя.

Пример 8. Передача одного параметра.

```
//STEP1 EXEC PGM=P1,PARM=10
```

Пример 9. Передача двух подпараметров.

```
//STEP2 EXEC PGM=IEWL,PARM='LIST,LET'
```

Пример 10. Передача подпараметров, содержащих специальные знаки.

```
//STEP3 EXEC PGM=ARIF,PARM=(LOAD,123,
//              'P1=5—B','A+B')
```

2.3. ОПИСАНИЕ НАБОРОВ ДАННЫХ

2.3.1. Общие сведения об операторе описания данных

Обрабатывающая программа, указанная оператором EXEC, использует определенные наборы данных: входные, выходные и промежуточные. Перечисленные наборы данных могут создаваться в этом пункте задания либо в предыдущих пунктах того же задания, либо в других заданиях. Все используемые обрабатывающей программой наборы данных должны быть описаны в этой программе и в задании. В задании каждый набор данных описывают оператором DD.

Набор данных, расположенный на магнитном носителе, может иметь стандартную метку. На диске стандартные метки обязательны, на магнитной ленте — не обязательны. При создании набора данных в стандартную метку записываются некоторые характеристики набора данных, указанные в программе и в операторе DD. Если в пункте задания используется существующий набор данных со стандартными метками, то сведения о нем, содержащиеся в метке, можно не повторять в программе и в задании.

В программе обычно указывают характеристики записей, из которых состоит набор данных, и особенности их обработки, а в операторе DD — местоположение и состояние набора данных, а также дополнительные характеристики, которые не были указаны в программе. Разделение описания набора данных между программой, заданием и меткой обеспечивает независимость программы от характеристик обрабатываемых наборов данных и упрощает программирование и составление заданий.

Во время выполнения программы все сведения о наборе данных из разных источников объединяются в *блоке управления данными,* представляющем собой таблицу, которая содержит харак-

теристики набора данных и используется программами управления данными ОС ЕС при выполнении операций ввода-вывода. Блок управления данными создается для каждого набора данных. Обычно он встроен в программу.

При программировании на языке Ассемблера блок управления данными строится и первоначально заполняется макрокомандой DCB (Data Control Block). Для программ на языках высокого уровня блоки управления данными создаются подпрограммами управления данными из библиотек компиляторов.

Чтобы получить доступ к набору данных при выполнении программы, нужно открыть блок управления данными. В программе на языке Ассемблера процедуру открытия вызывает макрокоманда OPEN (открыть), ссылающаяся на соответствующую макрокоманду DCB. В программах на языках высокого уровня для этого используется утверждение OPEN (например, в Коболе и ПЛ/1) или запрос на ввод-вывод, выдаваемый операторами ввода-вывода (в Алголе-60, Фортране и иногда в ПЛ/1).

Процедура открытия окончательно заполняет блок управления данными, используя информацию из соответствующего оператора DD, а также из метки набора данных (предварительно информация из оператора DD и метки объединяется в блоке управления файлом, создаваемом для каждого оператора DD). Заметим, что информация из оператора DD не может изменить поля блока управления данными, определенные в макрокоманде DCB, а информация из метки используется только в случае, если аналогичная информация не задана в макрокоманде DCB или в операторе DD.

Соответствие между набором данных, используемым в программе, и оператором DD, описывающим этот набор данных, устанавливается при помощи имени оператора DD. Его называют *dd-имя*. В программе на языке Ассемблера программист сам определяет *dd*-имя каждого используемого набора данных и указывает его в операнде DDNAME макрокоманды DCB. В программах на языках высокого уровня dd-имена наборов данных либо определяет программист (например, в Коболе, РПГ и ПЛ/1), либо они устанавливаются компилятором в соответствии с определенными соглашениями. (Эти соглашения описаны в п. 4.3.2.)

На рис. 2.2 показаны источники информации для блока управления данными при программировании на языке Ассемблера. Цифры у стрелок указывают очередность использования этих источников. Информация из оператора DD и метки набора данных предварительно объединяется в блоке управления файлом. (На рисунке он не показан.)

Оператор DD — наиболее сложный и наиболее употребительный оператор языка управления заданиями. В поле операндов этого оператора используют один позиционный параметр и двенадцать ключевых. Некоторые параметры содержат несколько подпараметров.

Наборы данных, описываемые оператором DD, могут находиться во входном потоке, выходном потоке и на внешних запоминаю-

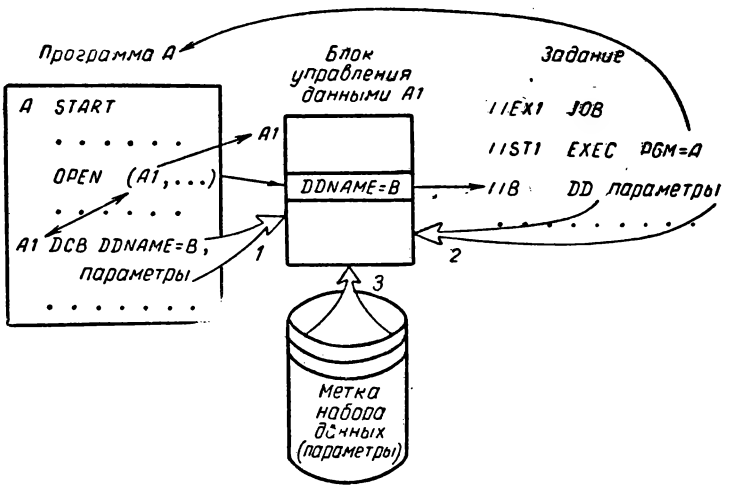


Рис. 2.2: Источники информации для блока управления данными

щих устройствах (устройствах ввода-вывода). Параметры, описывающие эти три группы наборов данных, несовместимы.

Во входном потоке могут находиться только входные наборы данных. Они вводятся через устройство системного ввода, через которое вводятся все задания системе. Обычно через входной поток вводят исходные данные обрабатывающих программ (в том числе исходные модули, подлежащие трансляции).

В выходной поток помещают выходные наборы данных, содержащие результаты решения задач и листинги системных обрабатывающих программ. Они выводятся через устройства системного вывода, общие для всех заданий.

На внешних магнитных запоминающих устройствах (магнитные ленты, диски и барабаны) могут располагаться входные, промежуточные и выходные наборы данных.

2.3.2. Имя оператора описания данных

Каждый оператор DD обычно должен иметь имя (dd-имя). Имя оператора DD используется для ссылок на этот оператор в программе, в управляющих операторах задания и в системных сообщениях. Имена операторов DD в задании должны соответствовать dd-именам наборов данных в программе. (Правила установления этого соответствия описаны в п. 4.3.2.) Программу можно транслировать и редактировать, не зная использованных в ней dd-имен наборов данных, но выполнить ее нельзя, если в задании не определены операторы DD с соответствующими dd-именами.

Иногда группа операторов DD может иметь одно dd-имя, записанное в поле имени первого оператора DD. В остальных опера-

торах группы поле имени должно быть пустым. Такие наборы данных называют *сцепленными* (соединенными). В обрабатывающей программе сцепленные наборы данных рассматриваются как один набор данных, поэтому сцепленные наборы данных должны иметь одинаковую организацию, а также одинаковую длину и формат записей.

Пример 1. Использование dd-имен.

```
//ST EXEC PGM=AB
//ABIN DD...
//ABT DD...
//     DD...
//ABOUT DD...
```

Обрабатывающая программа AB использует наборы данных, определенные операторами с dd-именами ABIN, ABT и ABOUT, причем dd-имя ABT относится к двум сцепленным наборам данных.

Некоторые dd-имена можно использовать только в определенном смысле. Как отмечалось в п. 2.2.2, для выполнения программы из общей системной библиотеки SYS1.LINKLIB достаточно записать имя этой программы в параметре PGM оператора EXEC. Чтобы сделать доступной в задании программу из личной библиотеки, сразу после оператора JOB нужно поместить оператор DD с dd-именем JOBLIB. Личная библиотека, указанная в этом операторе, доступна всем пунктам задания. В любом операторе EXEC задания можно указать программу из этой библиотеки, и она будет выполнена. Библиотеку, описанную оператором с dd-именем JOBLIB, называют *библиотекой задания.*

Программу из личной или системной библиотеки можно сделать доступной только в одном пункте задания. Для этого в пункте задания нужно описать *библиотеку пункта задания.* Эту библиотеку описывают в операторе с dd-именем STEPLIB. Использование библиотеки пункта задания отменяет для этого пункта библиотеку задания. Она становится недоступной.

В общем случае программа, указанная оператором EXEC, вначале отыскивается в библиотеке пункта задания, а если ее там не окажется, то поиск продолжается в общей системной библиотеке. Если библиотека пункта задания не определена, то поиск начинается с просмотра библиотеки задания и продолжается в общей системной библиотеке. Наконец, при отсутствии библиотек пункта задания и задания поиск начинается сразу в общей системной библиотеке.

Как для задания, так и для пункта задания можно определить несколько библиотек, задав их сцепленными операторами DD. (Пример использования в задании личных библиотек приведен в п. 2.3.6.)

Оператор DD с dd-именем SYSABEND или SYSUDUMP описывает набор данных, в который при аварийном завершении пункта задания записывается *дамп* (содержимое основной памяти и некоторая дополнительная информация о состоянии системы). (Дампы кратко описаны в разд. 5.1.) Подчеркнем еще раз, что

dd-имена JOBLIB, STEPLIB, SYSABEND и SYSUDUMP можно употреблять только в описанных здесь специальных операторах описания данных.

Имя оператора DD можно использовать для ссылок на этот оператор в том же или последующих пунктах задания. Ссылку записывают в одной из следующих форм:

```
*.dd-имя
*.имя пункта.dd-имя
*.имя пункта.имя пункта процедуры.dd-имя
```

Первую форму используют для ссылок на оператор DD в том же пункте задания, вторую — для ссылок на оператор DD в другом пункте, а третью — для ссылок на оператор DD в процедуре.

В дальнейшем для краткости ссылку, записанную в одной из приведенных форм, будем называть *ссылкой на dd-имя.*

Пример 2. Ссылка на dd-имя.

```
//GO EXEC PGM=*.LKED.SYSLMOD
```

Оператор EXEC предписывает выполнить загрузочный модуль, определенный в ранее выполненном пункте LKED оператором DD с именем SYSLMOD.

2.3.3. Наборы данных во входном и выходном потоках. Позиционный параметр и параметр SYSOUT

Позиционный параметр оператора DD имеет три значения:

```
*
DATA
DUMMY
```

Значение * или значение DATA указывает, что набор данных находится во входном потоке непосредственно за этим оператором DD. Конец набора данных во входном потоке отмечают оператором ограничения /*. В поле операндов оператора DD * или DD DATA не должно быть больше никаких параметров (кроме параметра DCB, о котором пойдет речь в п. 2.3.11).

При использовании значения позиционного параметра * вводимый набор данных не должен содержать управляющих карт (карт, содержащих знаки «//» или «/*» в первых двух колонках). Оператор DD * применяют для ввода через входной поток исходных данных обрабатывающей программы (например, для ввода исходного модуля при трансляции).

Значение DATA допускает присутствие во вводимом наборе данных управляющих карт, кроме карты оператора ограничения /*. Оператор DD DATA применяют для ввода процедур с целью последующей записи их в библиотеку (например, для каталогизации).

Пример 1. Набор данных во входном потоке, не содержащий управляющих карт.

```
//SYSIN DD *
набор данных на перфокартах, не содержащий знаков
// и /* в первых двух колонках каждой карты
/*
```

Пример 2. Набор данных во входном потоке, содержащий управляющие карты.

```
//SYSIN DD DATA
набор данных на перфокартах, не содержащий знаков /*
в первых двух колонках каждой карты
/*
```

В однопрограммном режиме PCP в каждом пункте задания допускается только один оператор DD * или DD DATA. Он должен быть последним оператором пункта задания. В мультипрограммных режимах такого ограничения нет.

Значение позиционного параметра DUMMY указывает, что набор данных фиктивный. Оператор DD DUMMY применяют в тех случаях, когда нужно отменить операции ввода-вывода, предусмотренные в обрабатывающей программе. Для фиктивного набора данных система не выделяет устройство. Например, если в обрабатывающей программе предусмотрены операции вывода с набором данных, имеющим dd-имя SYSPRINT, а в задании записан оператор //SYSPRINT DD DUMMY, то при назначении заданию устройств для набора данных SYSPRINT устройство выделено не будет, а при исполнении программы операции вывода на этот набор данных выполняться не будут. Заметим, что если в рассматриваемом примере просто удалить из задания оператор с dd-именем SYSPRINT, то задание закончится аварийно (не определен набор данных, требующийся обрабатывающей программе).

Параметр SYSOUT служит для определения набора данных в выходном потоке. Этот ключевой параметр указывает класс выходного потока. В простейших случаях применяют две формы параметра SYSOUT: SYSOUT=A или SYSOUT=B. Класс A — вывод на АЦПУ, а класс B — вывод на перфоратор. Подробнее параметр SYSOUT описан в приложении 1.

Пример 3. Наборы данных в выходном потоке:

```
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD DUMMY,SYSOUT=B
//SYSABEND DD SYSOUT=A
```

Оператор с dd-именем SYSPUNCH определяет фиктивный набор данных, а оператор с dd-именем SYSABEND предписывает в случае аварийного завершения пункта задания вывести на АЦПУ дамп, включающий область памяти выполняемой программы и область памяти ядра системы.

2.3.4. Наборы данных на внешних запоминающих устройствах

Для описания наборов данных на внешних запоминающих устройствах, к которым относятся магнитные ленты, диски и барабаны, используют ключевые параметры оператора DD. Основные ключевые параметры дают следующие сведения о наборе данных:

параметр DSNAME — имя набора данных и его вид (постоянный, временный);

параметр DISP — текущее состояние набора данных (существует, вновь создается) и состояние после выполнения пункта задания (сохраняется, каталогизируется, передается другим пунктам того же задания, уничтожается);

параметр UNIT — устройство (магнитная лента, диск или барабан);

параметр SPACE — требующийся объем памяти на диске и организация набора данных (последовательный, библиотечный и т. п.);

параметр VOLUME — том (номер бобины магнитной ленты, диска, барабана).

Остальные параметры задают дополнительные характеристики набора данных. Например, в параметре DCB можно указать длину записей, размер блоков, в которые объединяются записи, и другие характеристики записей, а в параметре LABEL — вид меток, срок сохранения набора данных и особенности его обработки. Имеется также группа параметров, обеспечивающих оптимизацию размещения наборов данных на устройствах (параметры SEP, AFF, SPLIT, SUBALLOC).

Состав параметров, используемых для описания конкретного набора данных, зависит от его состояния (существует или создается), вида (временный, сохраняемый, каталогизированный), организации (последовательный, библиотечный и т. п.) и устройства. Например, для выборки каталогизированного набора данных в большинстве случаев достаточно указать его имя и состояние, а для выборки сохраняемого набора данных дополнительно должно быть еще указано устройство и том. Это объясняется тем, что сведения об устройстве и томе для каталогизированного набора данных хранятся в каталоге системы.

В табл. 2.2 и 2.3 перечислены параметры оператора DD, используемые в типовых случаях при работе с последовательными и библиотечными наборами данных.

2.3.5. Параметр DSNAME

Параметр DSNAME (сокращенно DSN) определяет *имя набора данных*. Это имя называют «ds-имя» (data set name) в отличие от dd-имени — имени оператора DD. Имя должны иметь все постоянные наборы данных. Временному набору данных можно не давать имени.

Таблица 2.2

Параметры оператора DD, используемые при создании последовательных и библиотечных наборов данных

Вид набора данных	Обязательные параметры	Возможные дополнительные параметры
Временный набор данных на ленте	UNIT	DCB, VOLUME, LABEL, DISP
Временный набор данных на диске	UNIT, SPACE	DCB, VOLUME, LABEL, DISP
Временный набор данных в выходном потоке	SYSOUT	DCB, UNIT, SPACE
Постоянный набор данных на ленте	UNIT, DSNAME, DISP	DCB, VOLUME, LABEL
Постоянный последовательный набор данных на диске	UNIT, DSNAME, DISP, SPACE, (SPLIT, SUBALLOC)	DCB, VOLUME, LABEL
Постоянный библиотечный набор данных	UNIT, DSNAME, DISP, SPACE, (SUBALLOC)	VOLUME
Новый раздел библиотеки	DSNAME, DISP	UNIT, VOLUME

Таблица 2.3

Параметры оператора DD, используемые при выборке последовательных и библиотечных наборов данных

Вид набора данных	Обязательные параметры	Возможные дополнительные параметры
Каталогизированный набор данных	DSNAME, DISP	DCB, UNIT, LABEL
Сохраняемый набор данных на ленте	DSNAME, DISP, UNIT, VOLUME	DCB, LABEL
Сохраняемый последовательный набор данных на диске	DSNAME, DISP, UNIT, VOLUME	LABEL
Раздел библиотеки	DSNAME, DISP	UNIT, VOLUME
Набор данных, передаваемый из другого пункта задания	DSNAME, DISP	DCB, UNIT, VOLUME, LABEL

Имя постоянного последовательного набора данных или имя постоянной библиотеки записывают в виде:

DSNAME=имя

«Имя» может быть простым или составным. Простое имя состоит не более чем из восьми букв и цифр и начинается с буквы. Составное имя состоит из простых имен, разделенных точками. Оно не может иметь более 44 знаков, включая точки.

Для указания раздела библиотеки в операторе DD пишут:

DSNAME=имя библиотеки(имя раздела)

Перед именем временного набора данных (последовательного или библиотечного) записывают знаки && или один знак &. Отсутствие параметра DSNAME в операторе DD также означает, что набор данных временный. Если в описании набора данных параметр DSNAME опущен, то на этот набор данных можно сослаться в другом операторе DD, используя ссылку на dd-имя (см. п. 2.3.2).

2.3.6. Параметр DISP

Параметр DISP определяет состояние набора данных перед выполнением пункта задания и указывает, что делать с набором данных после завершения выполнения пункта задания. (Название этого параметра происходит от английского слова disposition, одно из значений которого — распоряжение.) Параметр DISP несовместим с параметром SYSOUT и является обязательным при создании постоянных наборов данных, а также при выборке наборов данных, расположенных на внешних магнитных запоминающих устройствах. Формат параметра:

DISP = (состояние, диспозиция, условная диспозиция)

Все подпараметры позиционные. Если опущены два последних подпараметра, то скобки не пишут.

Подпараметр «состояние» указывает состояние набора данных перед началом выполнения пункта задания и может иметь одно из четырех значений: NEW, OLD, MOD, SHR.

Значение NEW (новый) указывает, что набор данных создается в данном пункте задания. В этом случае необходимо также указать устройство, на котором создается набор данных. NEW является в системе стандартным значением, поэтому его можно явно не записывать.

Значение OLD (старый) показывает, что набор данных существует, и пункт задания получает монопольный доступ к нему. Если этот набор данных каталогизирован, то, кроме параметров DSNAME и DISP, больше никакие параметры могут не потребоваться.

Значение MOD (модифицируемый) указывает, что к набору данных будут добавляться записи, поэтому система установит механизм записи-считывания после последней записи. Для нового набора данных значения MOD и NEW эквивалентны.

Значение SHR (share — разделять) указывает, что набор данных существует и в режиме мультипрограммирования может одновременно использоваться для считывания несколькими заданиями. Обычно это значение записывают для системных библиотек (например, для библиотек компиляторов). В однопрограммном режиме значение SHR эквивалентно OLD.

Подпараметр «диспозиция» определяет, что делать с набором данных после нормального окончания пункта задания. Он может иметь одно из пяти значений: KEEP, DELETE, PASS, CATLG и

UNCATLG. Если состояние набора данных после выполнения пункта задания остается таким же, каким было до выполнения, то этот подпараметр можно опустить, при этом вновь созданный набор перестанет существовать, а старый набор, существовавший до начала выполнения пункта задания, сохранится. Отсюда следует, что параметр DISP можно не писать только для временных наборов данных, которые создаются и уничтожаются в одном пункте задания.

Значение KEEP (хранить) предписывает сохранить набор данных (но не каталогизировать его). Сохранить можно только постоянный набор данных (в параметре DSNAME для постоянного набора указано имя, которое не начинается знаками &). Для временных наборов данных значение KEEP эквивалентно значению PASS, а для каталогизированных наборов данных KEEP эквивалентно CATLG.

Значение DELETE (исключить) предписывает освободить память, занятую набором данных.

Значение PASS (передать) указывает, что набор данных передается последующим пунктам того же задания. Если в дальнейшем не будет указана окончательная диспозиция, то после завершения задания набор данных, созданный в задании (временный или постоянный), будет удален, а существующий постоянный набор данных будет продолжать существовать.

Значение CATLG (каталогизировать) предписывает каталогизировать набор данных. Каталогизировать можно только постоянный набор данных. При каталогизации имя набора данных (ds-имя) и сведения об устройстве и томе помещаются в каталог системы.

Значение UNCATLG предписывает удалить из каталога системы запись о наборе данных. Сам набор сохраняется.

Подпараметр «условная диспозиция» определяет, что делать с набором данных при аварийном завершении пункта задания. Он имеет те же значения, что подпараметр «диспозиция», кроме PASS. Если этот подпараметр опущен, то при аварийном завершении временные наборы данных уничтожаются, а постоянные — сохраняют прежнее состояние.

Пример. Использование в задании личных каталогизированных библиотек и наборов данных во входном и выходном потоках.

```
//EX1     JOB
//JOBLIB  DD    DSNAME=PLIB1,DISP=(OLD,PASS)
//        DD    DSNAME=PLIB2,DISP=(OLD,PASS)
//ST1     EXEC  PGM=P1
//PRINT   DD    SYSOUT=A
//IN      DD *
```

исходные данные программы P1

```
/*
//ST2       EXEC  PGM=PG
//STEPLIB   DD    DSNAME=LIBG, DISP=OLD
/*
```

```
//ST3      EXEC  PGM=IEWL
//STEPLIB  DD    DSNAME=SYS1.LINKLIB,DISP=OLD
. . . . . .
```

Библиотека задания (dd-имя JOBLIB) состоит из двух сцепленных личных библиотек PLIB1 и PLIB2, которые в свою очередь по умолчанию всегда сцеплены с общей системной библиотекой SYS1.LINKLIB. Предполагается, что обе личные библиотеки каталогизированы, поэтому для их вызова достаточно указания имени в параметре DSNAME и состояния OLD в параметре DISP. Диспозиция PASS указывает, что эти библиотеки передаются последующим пунктам задания.

Поиск программы P1, исполняемой в пункте ST1, происходит сначала в библиотеке PLIB1, если этой программы там не окажется, то поиск будет продолжен в библиотеке PLIB2, а затем — в общей системной библиотеке SYS1.LINKLIB. Поиск ведется до тех пор, пока не будет найден раздел библиотеки с именем P1, тогда поиск прекращается, а найденная программа загружается в основную память, настраивается по месту и выполняется. В ходе выполнения программа P1 получает исходные данные через входной поток (dd-имя набора данных IN) и выдает результаты через выходной поток на АЦПУ (dd-имя PRINT).

В пункте ST2 определена каталогизированная личная библиотека пункта задания LIBG (dd-имя STEPLIB), которая отменяет использование библиотеки задания. Библиотека пункта задания, как и библиотека задания в пункте ST1, сцеплена по умолчанию с общей системной библиотекой SYS1.LINKLIB, поэтому программа PG отыскивается сначала в библиотеке LIBG, а затем, если ее там не окажется, в общей системной библиотеке.

В пункте ST3 в качестве библиотеки пункта задания определена общая системная библиотека SYS1.LINKLIB. Это отменяет библиотеку задания. Программа IEWL будет отыскиваться в SYS1.LINKLIB.

2.3.7. Параметр UNIT

При создании новых наборов данных и выборке сохраняемых наборов данных нужно указывать устройство. Для этого служит параметр UNIT (буквально «устройство»). В простейшем случае параметр UNIT записывают в формате:

UNIT=(устройство,число устройств)

Подпараметр «устройство» указывает либо адрес, либо типовой номер, либо групповое имя устройства.

Адрес — трехразрядное шестнадцатеричное число, состоящее из номера канала, номера устройства управления и номера устройства. Например, 192 означает канал 1, устройство управления 9, устройство 2. В мультипрограммных режимах использование адреса для идентификации устройства может привести к конфликтам запросов разных заданий и задержкам выполнения заданий, поэтому адресом пользуются редко.

Типовой номер—номер, под которым объединяются однотипные устройства, имеющие разные коды. Например, 5Ø5Ø — типовой номер совместимых накопителей на магнитных дисках ЕС-5050, ЕС-5052 и ЕС-5056, а 7Ø3Ø — типовой номер АЦПУ ЕС-7030, ЕС-7032 и др.

Групповое имя (иногда говорят имя класса) — имя группы устройств, используемое в операционной системе. В частности, в ОС ЕС приняты такие групповые имена: SYSCP — выходные перфораторы карт, SYSSQ — магнитные ленты и диски (барабаны), SYSDA — магнитные диски (барабаны).

Типовые номера и групповые имена устанавливают при генерации системы. В конкретных вариантах ОС ЕС, помимо указанных выше групповых имен, могут использоваться также дополнительные групповые имена, например TAPE, — для магнитных лент.

При указании устройства типовым номером или групповым именем система сама выбирает устройство указанного типа (группы).

«Число устройств» — десятичное число. При назначении одного устройства число устройств не указывают, а скобки опускают. Например, параметр UNIT=5050 запрашивает для размещения набора данных диск, а UNIT=SYSSQ требует магнитную ленту или диск (барабан). В обоих случаях система сама выбирает одно устройство.

Дополнительные сведения о параметре UNIT приведены в п. 2.3.12 и приложении 1.

2.3.8. Параметр SPACE

При создании набора данных на устройстве прямого доступа (диске, барабане) нужно указать требуемый объем памяти, потому что на диске (барабане) система обычно размещает несколько наборов данных, и для распределения устройств система должна иметь сведения об объеме памяти, требуемом каждым набором данных. Для указания объема памяти служит параметр SPACE (пространство, область). Чаще всего этот параметр используют в форме:

SPACE=(единицы,(количество,приращение,оглавление),RLSE)

Компоненты «приращение» и «оглавление» второго подпараметра, а также подпараметр RLSE не обязательны, иногда их можно не указывать.

Подпараметр «единицы» определяет, в каких единицах запрашивается память. Он может иметь одно из трех значений: TRK — память определяется в дорожках (track — дорожка); CYL — память определяется в цилиндрах (напомним, что такие распространенные диски, как ЕС-5056 и ЕС-5052 имеют по 200 основных цилиндров, а каждый цилиндр состоит из десяти дорожек по 3625 байт); «размер блока» — память определяется в блоках, причем значение «размер блока» — десятичное число, равное среднему размеру блока записей в байтах.

Компоненты второго подпараметра (заключенного в круглые скобки) — десятичные числа, имеющие следующий смысл: «количество» — первоначальное количество единиц памяти, которое нужно выделить; «приращение» — количество единиц памяти, выделяемое дополнительно, если первоначальное количество окажется недостаточным (дополнительная память выделяется до 15 раз); «оглавление» (иногда говорят «справочник») — количество блоков оглавления библиотеки.

При распределении устройств для размещения наборов данных система учитывает только первоначальное количество запрошенной памяти, поэтому в ходе выполнения программы для памяти, выделяемой дополнительно («приращение»), может не хватить места на диске. В этом случае нужно увеличить в параметре SPACE запрос на начальное количество памяти, а затем — повторить выполнение задания (система выделит другой диск).

Компоненту «оглавление» записывают только для библиотечных наборов данных. Отсутствие этой компоненты означает, что набор данных последовательный. Следует иметь в виду, что память для оглавления библиотеки отводится за счет первоначальной памяти, выделенной набору данных, а каждый блок оглавления занимает 256 байт и содержит несколько записей о разделах библиотеки (размер каждой записи от 14 до 76 байт). Память для оглавления назначается при создании библиотечного набора данных. В дальнейшем ее расширить нельзя.

Подпараметр RLSE (release — освобождать) указывает, что после завершения формирования набора данных вся выделенная ему, но неиспользованная память освобождается.

Пример. Создание временных наборов данных.

```
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(100,10))
//SYSLMOD  DD  DSNAME=&&GOSET(MAIN),DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(1024,(20,10,1),RLSE)
```

Первый оператор предписывает создать временный (нет параметра DSNAME) последовательный (нет компоненты «оглавление» во втором подпараметре параметра SPACE) набор данных на диске (UNIT=SYSDA). Для этого набора данных первоначально запрашивается 100 дорожек, приращение 10 дорожек.

Второй оператор (dd-имя SYSLMOD) предписывает создать временную библиотеку (имя библиотеки начинается знаками &&) на диске, передаваемую (есть подпараметр PASS) последующим пунктам задания. Память запрашивается в блоках по 1024 байт. В оглавлении будет только один блок (256 байт). После создания библиотеки оставшаяся свободной память будет освобождена (есть подпараметр RLSE).

2.3.9. Параметр VOLUME

Часть внешнего запоминающего устройства, обслуживаемую одним механизмом доступа, называют томом. *Том* — это бобина магнитной ленты, дисковый пакет или магнитный барабан. На конкретной машине каждый том имеет уникальный *серийный номер*, представляющий собой произвольную комбинацию не более шести знаков ДКОИ. Обычно используют буквы и цифры. Серийный номер наносят на внешней поверхности тома так, чтобы его можно было видеть, и записывают в метке тома. Это позволяет обслуживающему персоналу устанавливать на устройства тома с заданными серийными номерами. Система, считывая метки, может проверять номера установленных томов и может запрашивать установку томов с определенными номерами, если их нет на устройствах.

Для создания набора данных на конкретном томе и для выборки набора данных с определенного тома используют параметр VOLUME (том), сокращенно VOL. Запрос конкретного тома называют *специальным* в отличие от *неспециального* запроса произвольного тома.

Параметр VOLUME обязателен при выборке сохраняемых наборов данных, созданных в одном задании, а используемых в другом. Для каталогизированных наборов данных информация о томах хранится в каталоге системы, поэтому при выборке каталогизированного набора данных том указывать не нужно, кроме того случая, когда набор данных многотомный и требуется начать обработку с конкретного тома.

Обычно параметр VOLUME записывают в таком формате:

VOLUME=(PRIVATE,RETAIN,номер тома,число томов,
SER=(список серийных номеров))

Первые четыре подпараметра позиционные. Подпараметр SER — ключевой. Любой подпараметр можно опустить (но не все сразу).

Подпараметр PRIVATE (личный) объявляет том *личным*. Личный том используется только для одного набора данных. После завершения пункта задания система подготавливает такой том к снятию с устройства. В последующем другие наборы данных можно разместить на этом томе только по специальному запросу.

Подпараметр RETAIN (сохранить) предписывает сохранить личный (или любой другой) том на устройстве при переходе от одного пункта задания к другому, если это возможно. Невозможность сохранения тома может быть вызвана нехваткой устройств для выполнения задания.

Подпараметр «номер тома» — десятичное число, указывающее порядковый номер тома многотомного набора данных, с которого начинается обработка набора данных.

Подпараметр «число томов» — десятичное число, указывающее число томов при неспециальном запросе многотомного набора данных. Этот подпараметр несовместим с ключевым подпараметром SER.

Подпараметр SER (serial — серийный) при специальном запросе задает серийные номера томов. Если запрашивается только один том, то значение этого подпараметра записывают без скобок. Вместо подпараметра SER можно также записать подпараметр REF, определяющий ссылку на том, занимаемый ранее определенным набором данных. Ссылку на том, содержащий каталогизированный или передаваемый из другого пункта задания набор данных, записывают в форме

REF=ds-имя

Если набор данных, определенный в том же задании, не каталогизирован и не передается, то ссылку на занимаемый им том записывают в форме

REF=ссылка на dd-имя

Ссылка на dd-имя записывается по общим правилам (см. п. 2.3.2).

Пример 1. Создание и сохранение набора данных на магнитной ленте.

```
//OUTPUT DD DSNAME =BETA,UNIT=5Ø1Ø,DISP=(,KEEP),
//          VOLUME=SER=T112
```

Пример 2. Создание и каталогизация набора данных на диске.

```
//OUTPUT2 DD DSN=A.B1,VOL=REF=*.ST1.OUTPUT1,DISP =(,CATLG),
//             SPACE=(CYL,(2,1))
```

Для набора данных с ds-именем A.B1 запрашивается том, на котором расположен набор данных с dd-именем OUTPUT1, определенный в пункте ST1.

Пример 3. Выборка сохраняемого набора данных с магнитной ленты.

```
//INPUT DD DSN=BETA,UNIT =5Ø1Ø,DISP=(OLD,KEEP),VOL=SER=T112
```

Вызывается набор данных, созданный в примере 1.

2.3.10. Параметр LABEL

В некоторых случаях в операторе DD записывают параметр LABEL (метка), который указывает порядковый номер набора данных на магнитной ленте, а также тип меток, наличие пароля, защищающего набор данных от несанкционированного считывания или изменения, особенности обработки набора данных и срок его сохранения. Параметр LABEL имеет формат:

LABEL=(номер, тип меток, вид защиты, вид обработки, срок хранения)

Первые четыре подпараметра позиционные, а последний подпараметр ключевой.

Подпараметр «номер» — десятичное число, указывающее позицию набора данных на магнитной ленте по отношению к другим наборам данных в томе или группе томов (для многотомных наборов данных).

Подпараметр «тип меток» определяет тип меток (стандартные, нестандартные и т. п.). Пропуск этого подпараметра означает, что метки стандартны.

Подпараметр «вид защиты» имеет два значения: PASSWORD и NOPWREAD. Значение PASSWORD (пароль) запрещает доступ к набору данных, если оператор в ответ на запрос системы не укажет правильный пароль. Значение NOPWREAD (читать без пароля) позволяет считывать набор данных без пароля, но не позволяет изменить или исключить этот набор данных, если оператор не назовет пароль. Пропуск этого подпараметра означает, что защита не предусмотрена.

Подпараметр «вид обработки» имеет два значения IN и OUT. Значение IN указывает, что набор данных используется только

для ввода, а OUT — только для вывода. Попытка использовать набор данных не так, как указано, рассматривается системой как ошибка.

Ключевой подпараметр «срок хранения» определяет либо период хранения набора данных, либо дату, до которой он должен храниться.

Пример. Параметр LABEL для набора данных со стандартными метками, защищенного от изменений и используемого только для вывода.

LABEL=(„NOPWREAD,OUT)

2.3.11. Параметр DCB

Как было отмечено в п. 2.3.1, каждый набор данных, используемый в обрабатывающей программе, должен иметь блок управления данными. Этот блок может быть не полностью определен в обрабатывающей программе. Некоторые недостающие сведения можно указать в параметре DCB оператора DD. В отличие от других параметров оператора DD, которые определяют в основном местоположение набора данных и его состояние, параметр DCB указывает сведения, необходимые для обработки записей, из которых состоит набор данных. Обычно оператор DCB записывают в таком формате:

DCB=(список подпараметров)

Всего имеется 21 подпараметр, все они ключевые, некоторые из них являются взаимоисключающими.

Подпараметры параметра DCB могут указывать такие сведения:

1) организацию набора данных и особенности доступа к записям (подпараметры DSORG, EROPT, LIMCT, NCP, NTM, OPTCD);

2) характеристики блоков и записей (подпараметры BLKSIZE, KEYLEN, LRECL, RECFM, RKP);

3) характеристики буферов, используемых при вводе-выводе (подпараметры BFALN, BFTEK, BUFL, BUFNO);

4) характеристики размещения и обработки записей, определяемые особенностями устройств (подпараметры CYLOFL, DEN, MODE, PRTSP, STACK).

Чаще других используют подпараметры «BLKSIZE=размер блока», «BUFNO=число буферов» и «LRECL=длина записи». Первый подпараметр задает длину блока записей в байтах, второй — число буферов, а третий — длину логической записи.

Для краткости вместо выписывания списка подпараметров можно записать параметр DCB в одной из следующих форм:

DCB=ds-имя
DCB=ссылка на dd-имя

Первая форма предписывает скопировать значения подпараметров параметра DCB из метки каталогизированного набора дан-

ных с указанным ds-именем, а вторая — из метки набора данных, определенного ранее в том же задании оператором DD с заданным dd-именем. При копировании подпараметров из метки после имени каталогизированного набора данных или после ссылки на dd-имя можно указать список подпараметров, значения которых не копируются, а определяются непосредственно в параметре DCB.

Пример. Параметр DCB, копирующий подпараметры из метки набора данных, определенного ранее в том же задании.

```
DCB=(*.ST1.SYSUT3,BLKSIZE=400)
```

2.3.12. Оптимизация размещения наборов данных

В языке управления заданиями имеются средства, позволяющие управлять размещением наборов данных на устройствах с целью сокращения времени выполнения операций ввода-вывода или экономии числа занятых устройств. К таким средствам относятся подпараметры SEP и AFF параметра UNIT, а также параметры оператора DD — SEP, AFF, SPLIT и SUBALLOC.

Ключевой подпараметр SEP (separate — отделять) параметра UNIT служит для сокращения времени выполнения операций ввода-вывода за счет размещения наборов данных на отдельных устройствах прямого доступа. Подпараметр

```
SEP=(список dd-имен)
```

указывает, что определяемый набор данных не следует по возможности размещать на тех устройствах, где размещены наборы данных, dd-имена которых перечислены в списке.

Пример 1. Назначение устройств с использованием подпараметра SEP.

```
//SYSUT1   DD   UNIT=SYSSQ,SPACE=(1700,  (400,50))
//SYSUT2   DD   UNIT=SYSSQ,SPACE=(1700,  (400,50))
//SYSUT3   DD   UNIT=(SYSSQ,SEP=(SYSUT1,SYSUT2)),
//              SPACE=(1700,  (400,50))
```

Ключевой подпараметр AFF (affix — присоединить) служит для экономии числа устройств, используемых обрабатывающей программой. Он запрашивает устройство общее с другим набором данных. Параметр UNIT в форме

```
UNIT=AFF=dd-имя
```

предписывает разместить набор данных на том же устройстве, что и набор данных, определенный оператором DD с указанным dd-именем. Заметим, что подпараметр AFF несовместим с другими подпараметрами параметра UNIT.

В ЕС ЭВМ каждое устройство ввода-вывода подключено к определенному каналу. При описании наборов данных можно оптимизировать не только использование устройств, но и использование каналов. Для этого предназначены параметры SEP и AFF. Подчеркнем, что здесь речь идет не о подпараметрах параметра UNIT, а о самостоятельных параметрах.

Параметр SEP, записываемый в форме

SEP=(список dd-имен)

указывает, что определяемый набор данных должен использовать канал, отличный от каналов, используемых наборами данных, dd-имена которых перечислены в списке.

Параметр AFF, имеющий форму

AFF=dd-имя

ссылается на оператор DD, в котором записан параметр SEP, и запрашивает такое же отделение каналов. Иными словами, вместо AFF можно подставить параметр SEP, записанный в операторе, на который ссылается AFF. Эффект будет тот же. Следовательно, параметр AFF выполняет роль своеобразного «местоимения» параметра SEP.

Пример 2. Использование параметров SEP и AFF.

```
//SYSUT1 DD DSN=&UT1,UNIT=SYSSQ,SPACE=(1Ø24,(50,1Ø))
//SYSUT2 DD UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1Ø24,(5Ø,1Ø))
//SYSUT3 DD UNIT=SYSSQ,AFF=SYSUT2,SPACE=(1Ø24,(50,1Ø))
```

Если каналов достаточно, то наборы данных, определенные операторами с dd-именами SYSUT2 и SYSUT3, будут приписаны к каналам, которые не совпадают с каналом, используемым набором данных &UT1.

Подпараметры и параметры SEP и AFF ограничивают использование устройств и каналов. Это может привести к выдаче сообщений, на которые должен реагировать оператор, и к ненужному демонтированию томов (подготовке томов к снятию с устройств). Поэтому эти средства целесообразно применять только в тех случаях, когда устройств и каналов достаточно. На некоторых машинах, работающих в мультипрограммном режиме, применение подпараметра и параметра SEP запрещают в административном порядке.

Параметр SPACE также имеет два подпараметра, которые служат для ускорения выполнения операций ввода-вывода. Эти позиционные подпараметры записывают после подпараметра RLSE (см. п. 2.3.8).

Первый позиционный подпараметр, записываемый вслед за подпараметром RLSE, требует, чтобы память на диске выделялась сплошным участком. Это объясняется тем, что при выделении первоначальной памяти, запрошенной параметром SPACE, система может назначить до пяти несмежных участков. Это замедляет выполнение операции ввода-вывода, хотя и улучшает использование свободного пространства на диске. Ввод-вывод происходит быстрее, если набор данных занимает сплошной непрерывный участок памяти. Описываемый подпараметр может иметь одно из трех значений: CONTIG, MXIG и ALX. Значение CONTIG требует, чтобы заказанная память выделялась одним сплошным участком. Значение MXIG указывает, что для набора данных нужно выделить наибольшую сплошную свободную область из числа имею-

щихся на томе, размер которой не меньше указанного в параметре SPACE. Значение ALX предписывает выделить до пяти сплошных областей, каждая из которых не меньше указанной в параметре SPACE величины.

Второй подпараметр имеет одно значение ROUND и требует, чтобы размер памяти, указанной в блоках, был округлен до целого числа цилиндров (это ускоряет ввод-вывод).

Напомним, что все подпараметры параметра SPACE позиционные, поэтому пропуск подпараметра отмечается запятой, кроме случая, когда опущенные подпараметры последние в списке позиционных подпараметров.

Пример 3. Запрос сплошной области на диске, округленной до целого числа цилиндров.

```
//SYSUT DD UNIT=SYSDA,SPACE =(1024,(40,10),,CONTIG,ROUND)
```

Параметр SPLIT (разделять) используют в тех случаях, когда несколько последовательных, размещенных на одном диске наборов данных нужно обрабатывать одновременно. Для сокращения времени доступа к таким наборам данных целесообразно разместить их на одних и тех же цилиндрах так, чтобы каждый набор данных занимал часть дорожек цилиндра (обычно цилиндр состоит из десяти дорожек). Это уменьшит количество перемещений головок записи-считывания.

Операторы DD для наборов данных, разделяющих цилиндры, записывают подряд. В этих операторах память запрашивают в цилиндрах или в блоках. Если память запрошена в цилиндрах, то в первом операторе DD параметр SPLIT записывают в виде

SPLIT=(n,CYL,(величина,приращение))

Здесь n — количество дорожек на каждом цилиндре, выделяемое первому набору данных; «величина» — первоначальное количество цилиндров, выделяемое для всех связанных наборов данных; «приращение» — дополнительное число цилиндров, выделяемое, если хотя бы один набор данных исчерпает выделенную ему память. Дополнительная память используется без разделения цилиндров. «Приращение» — необязательный подпараметр.

В каждом следующем операторе DD параметр SPLIT имеет вид

SPLIT=n

Здесь n — число дорожек на каждом цилиндре, выделяемое данному набору данных.

Пример 4. Наборы данных, разделяющие цилиндры.

```
//SYSUT1 DD UNIT=SYSDA,SPLIT=(2,CYL,(40,10))
//SYSUT2 DD UNIT=SYSDA,SPLIT=4
//SYSUT3 DD UNIT=SYSDA,SPLIT=4
```

Если память запрашивается в блоках, то в первом операторе DD записывают

SPLIT=(%,размер блока,(величина, приращение))

а в последующих

SPLIT=%

Здесь % — число, определяющее процент дорожек на каждом цилиндре, выделяемых соответствующему набору данных: «размер блока» — средний размер блока в байтах; «величина» — общее число блоков для всех связанных наборов данных; необязательный подпараметр «приращение» — дополнительное количество блоков, выделяемое при нехватке первоначально выделенной памяти.

Система переводит запрос памяти в блоках в целое число цилиндров и после этого распределяет память на диске.

Пример 5. Запрос памяти в блоках для наборов данных, разделяющих цилиндры:

```
//SYSUT1 DD UNIT=SYSDA,SPLIT=(40,1700,(200,50))
//SYSUT2 DD UNIT=SYSDA,SPLIT=60
```

Параметр SUBALLOC (подраспределение) используют совместно с параметром SPACE в тех случаях, когда на одном томе прямого доступа (диске) нужно разместить в определенной последовательности несколько наборов данных. Вначале записывают оператор DD с параметром SPACE, который содержит подпараметр CONTIG и запрашивает память, достаточную для размещения всех наборов данных. Затем для каждого создаваемого набора данных записывают оператор DD с параметром SUBALLOC. Этот параметр имеет формат:

```
SUBALLOC=(единицы,(количество, приращение, оглавление),
                  ссылка на dd-имя)
```

Здесь все подпараметры, кроме «ссылки на dd-имя», имеют тот же смысл и те же значения, что в параметре SPACE (см. п. 2.3.8). «Ссылка на dd-имя» указывает оператор DD, в котором определена память для всех наборов данных. Для определяемого набора данных память выделяется из этой общей памяти. Дополнительная память, которую определяет необязательный подпараметр «приращение», не входит в общую память. Подпараметр «оглавление» нужен только для библиотечных наборов данных (см. п. 2.3.8).

Пример 6. Использование подраспределения памяти на диске.

```
//A DD DSN=A,UNIT=SYSDA,SPACE=(CYL,50,,CONTIG)
//B DD DSN=B,SUBALLOC=(TRK,200,A)
//C DD DSN=C,SUBALLOC=(CYL,25,A)
```

Заметим, что в отличие от обычной формы ссылки на dd-имя в параметре SUBALLOC эта ссылка записывается без знаков «*.» в начале ссылки.

2.4.1. Описание процедуры

Процедура представляет собой последовательность управляющих операторов языка управления заданиями, которую можно вызвать по имени и использовать в пункте задания. Для вызова процедуры достаточно в качестве позиционного параметра оператора EXEC записать «PROC=имя процедуры» или просто «имя процедуры» (см. п. 2.2.2).

В зависимости от места, где находится процедура, различают процедуры во входном потоке и каталогизированные процедуры. Процедура во входном потоке записана в задании и доступна только в этом задании. Каталогизированная процедура находится в системной библиотеке процедур SYS1.PROCLIB и доступна в любом задании.

Процедуры во входном потоке обычно помещаются в начале задания непосредственно после оператора JOB или после оператора описания данных с dd-именем JOBLIB. Каждая такая процедура начинается оператором процедуры PROC и заканчивается оператором конца процедуры PEND. Формат этих операторов приведен в табл. 2.1. В задании можно описать до 15 процедур.

Каждая процедура должна иметь уникальное имя. Для процедуры во входном потоке имя записывают в поле имени оператора PROC.

Каталогизированные процедуры в отличие от процедур во входном потоке не содержат оператора PEND. Оператор PROC записывают в начале каталогизированной процедуры лишь в том случае, когда нужно задать стандартные значения символических параметров, о которых будет сказано ниже. В библиотеку процедур SYS1.PROCLIB каталогизированную процедуру помещает специальная утилита (вспомогательная системная программа) IEBUPDTE. Имя каталогизированной процедуры есть имя содержащего ее раздела библиотеки.

Управляющие операторы любой процедуры могут быть только операторами EXEC и DD или операторами комментария, причем операторы EXEC не могут обращаться к другим процедурам, а среди операторов DD не должно быть операторов с dd-именем JOBLIB и операторов с параметром * или DATA. Каждая процедура должна содержать по крайней мере один оператор EXEC.

При вызове процедуры значения параметров в ее управляющих операторах можно изменить. Для упрощения замены параметров используют символические параметры.

Символический параметр — это имя, начинающееся с буквы и содержащее не более семи латинских букв, цифр и знаков #, @, $. В поле операндов операторов процедуры перед символическим параметром записывают знак &. Например UNIT=&TYPE, DSN=&NAME. Здесь TYPE и NAME — символические параметры.

При вызове процедуры символические параметры должны получить значения (быть может пустые). Эти значения указывают в операторах PROC и EXEC.

Оператор PROC задает стандартные значения символических параметров. Список символических параметров и их стандартных значений записывают в поле операндов оператора PROC. Каждый элемент списка имеет форму

символический параметр=значение

Элементы в списке разделяют запятыми. Подчеркнем, что в операторе PROC символический параметр записывают без предшествующего знака &.

Оператор EXEC, вызывающий процедуру, указывает значения тех символических параметров, значения которых отличаются от стандартных, и тех, для которых не предусмотрены стандартные значения. Список значений символических параметров в операторе EXEC записывают по тем же правилам, что список ключевых параметров этого оператора, поэтому символические параметры должны отличаться от ключевых слов оператора EXEC. Если символическому параметру присваивается пустое значение в виде «символический параметр= », то это значит, что соответствующий параметр нужно опустить.

Пример 1. Описание процедуры во входном потоке.

```
//PR       PROC TYPE=SYSSQ,PRINT=A
//ST1      EXEC PGM=B
//STEPLIB  DD   DSN=LIB1,DISP=OLD
//SYSIN    DD   DSN=&NAME,DISP=OLD
//UT       DD   UNIT=&TYPE,SPACE=(TRK,200)
//SYSOUT   DD   SYSOUT=&PRINT
//        PEND
```

Эта процедура имеет имя PR, записанное в поле имени оператора PROC. В процедуре используются три символических параметра: NAME, TYPE и PRINT. Параметрам TYPE и PRINT в операторе PROC назначаются стандартные значения. Параметр NAME не имеет стандартного значения, поэтому значение ему нужно присвоить в операторе EXEC, вызывающем процедуру.

Если в задании, содержащем описание процедуры PR, написан оператор

```
//ST EXEC PR,NAME=B1.A2,TYPE=5050
```

то это равносильно такой последовательности операторов:

```
//ST1      EXEC PGM=B
//STEPLIB  DD   DSN=LIB1,DISP=OLD
//SYSIN    DD   DSN=B1.A2,DISP=OLD
//UT       DD   UNIT=5050,SPACE=(TRK,200)
//SYSOUT   DD   SYSOUT=A
```

Здесь параметры NAME и TYPE получили значения, указаные в операторе EXEC, а параметр PRINT — стандартное значение.

Заметим, что назначенные оператором EXEC значения символических параметров действуют только в текущем пункте задания. В общем случае несколько символических параметров можно

соединить вместе. Допускается также объединение символического параметра с постоянной информацией. Детально правила использования символических параметров описаны в эксплуатационной документации по ОС ЕС.

Иногда обрабатывающая программа, указанная в процедуре, должна получить исходные данные из входного потока. Но использование в процедуре операторов вида DD * и DD DATA недопустимо. Чтобы обойти это ограничение, в процедуре записывают оператор DD с параметром DDNAME=dd-имя, который указывает dd-имя оператора, определяющего требуемые исходные данные во входном потоке. Иными словами, параметр DDNAME позволяет «отложить» определение набора данных до того момента, когда среди управляющих операторов задания появится оператор с dd-именем, заданным в качестве значения параметра DDNAME. Если нужно ввести данные через входной поток, то оператор во входном потоке, на который ссылается параметр DDNAME, записывают в виде

```
//имя пункта процедуры.dd-имя DD *
```

Здесь «имя пункта процедуры» показывает, к какому пункту процедуры нужно присоединить этот оператор DD (в процедуре может быть несколько пунктов), а «dd-имя» совпадает с dd-именем, записанным в параметре DDNAME.

Пример 2. Использование параметра DDNAME.

```
//PR1      PROC
//STEP1    EXEC PGM=A1
//INPUT    DD   DDNAME=IN
//SYSOUT   DD   SYSOUT=A
//         PEND
```

Предполагается, что обрабатывающая программа A1 должна получить исходные данные в наборе данных с dd-именем INPUT. Параметр DDNAME=IN откладывает определение этого набора данных до тех пор, пока во входном потоке не встретится оператор с dd-именем вида STEP1.IN.

Пример 3. Вызов процедуры PR1 из примера 2, получающей исходные данные через входной поток.

```
//STEPA EXEC PR1
//STEP1.IN DD *
карты с исходными данными программы A1
/*
```

Выполнение этих операторов равносильно выполнению операторов:

```
//STEP1  EXEC PGM=A1
//SYSOUT DD   SYSOUT=A
//INPUT  DD   *
карты с исходными данными программы A1
/*
```

Параметр DDNAME можно использовать не только для ввода данных через входной поток. При помощи этого параметра к процедуре можно присоединить набор данных, расположенный на любом устройстве.

Пример 4. Вызов процедуры PR1 из примера 2, получающей исходные данные из каталогизированного набора данных

```
//STEPB EXEC PR1
//STEP1.IN DD DSNAME =DATA1,DISP=OLD
```

Взаимодействие оператора с dd-именем INPUT в процедуре и оператора с dd-именем STEP1.IN равносильно определению в процедуре оператора

```
//INPUT DD DSNAME=DATA1,DISP=OLD
```

определяющего каталогизированный набор данных DATA1

2.4.2. Замещение и добавление параметров

Управляющие операторы в процедуре можно модифицировать и без использования символических параметров. Средства языка управления заданиями позволяют при вызове процедуры изменять и добавлять параметры в операторах EXEC и операторах DD процедуры, а также добавлять к процедуре дополнительные операторы DD. Операторы EXEC добавлять к процедуре нельзя.

Изменения процедуры действуют только в текущем пункте задания, в котором процедура вызывается. Сама процедура (во входном потоке или в библиотеке процедур) остается неизменной.

Для модификации операндов оператора EXEC в процедуре нужно в операторе EXEC, вызывающем процедуру, для каждого изменяемого или добавляемого параметра записать новое значение следующим образом:

ключевое слово.имя пункта процедуры=значение (2.1)

Здесь «ключевое слово» — ключевое слово изменяемого или добавляемого параметра, а «имя пункта процедуры» — имя модифицируемого оператора EXEC. Новое значение параметра нужно записывать полностью. В частности, если изменяется только один подпараметр какого-либо параметра, то нужно выписать явно все подпараметры, включая и те, которые сохраняют прежние значения.

Пример 1. Модификация и добавление параметров в операторе EXEC процедуры.

Требуется при вызове каталогизированной процедуры LKED, содержащей оператор

```
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET,NCAL',REGION=96K
```

устранить в параметре PARM подпараметры XREF,LET,NCAL и добавить подпараметр MAP, подпараметр LIST сохраняется (все подпараметры ключевые). Кроме того, нужно добавить параметр COND=(8,LE,ST1).

Решение дает следующий оператор:

```
//ST2 EXEC LKED,PARM.LKED='MAP,LIST',COND.LKED=(8,LE,ST1)
```

Процедура может состоять из нескольких пунктов (содержать несколько операторов EXEC). Используя записи вида (2.1), можно изменить несколько параметров в каждом операторе EXEC процедуры. Для этого в операторе EXEC, вызывающем процедуру, сначала записывают все изменения для первого пункта процедуры (первого оператора EXEC), затем для второго и т. д. В том случае, когда какому-либо параметру во всех операторах EXEC нужно назначить одинаковые значения, в операторе EXEC, вызывающем процедуру, записывают этот параметр без указания имени пункта процедуры в виде

ключевое слово=значение

Заметим, что записанный так параметр PARM модифицирует только первый пункт процедуры и ликвидирует параметр PARM во всех других пунктах этой процедуры, а параметр TIME имеет значение для всей процедуры.

Пример 2. Модификация параметров в нескольких операторах EXEC процедуры.

Если в процедуре PREX два пункта:

```
//ST1 EXEC PGM=A,PARM=1
.
.
.
//ST2 EXEC PGM=B,PARM=2,COND=(12,LE,ST1)
```

то вызов

```
//STA EXEC PREX,PARM.ST1=,PARM.ST2=1,COND.ST2=EVEN
```

равносилен выполнению операторов

```
//ST1 EXEC PGM=A
.
.
.
//ST2 EXEC PGM=B,PARM=1,COND=EVEN
```

а вызов

```
//STB EXEC PREX,PARM=2,COND=
```

равносилен выполнению операторов

```
//ST1 EXEC PGM=A,PARM=2
.
.
.
//ST2 EXEC PGM=B
```

В рассмотренном примере использована запись параметров с пустыми значениями (например, COND=). Такая запись уничтожает модифицируемый параметр.

Правила модификации операторов DD отличаются от правил изменения параметров в операторе EXEC. Модифицируемые операторы DD записывают вслед за оператором EXEC, вызывающим процедуру, в таком формате:

```
//имя пункта процедуры.dd-имя DD операнды          (2.2)
```

Здесь «имя пункта процедуры» называет пункт процедуры, а «dd-имя» указывает модифицируемый оператор DD в этом пункте. Для первого пункта процедуры «имя пункта процедуры» можно опустить. В поле операндов записывают только те параметры, которые нужно изменить или добавить. Все подпараметры изменяемого параметра DD выписывают явно, включая те, которые сохраняют прежние значения. Последнее не относится к парамет-

ру DCB. В этом параметре указывают только изменяемые и добавляемые подпараметры.

Для удаления параметра записывают его с пустым значением. Например, «DSN=» удаляет параметр DSNAME. Параметр удаляется также в том случае, когда указан исключающий его параметр. Например, если в операторе DD имеется параметр SPACE, а добавляется параметр SPLIT, то параметр SPACE будет удален. Чтобы удалить параметр DCB, нужно удалить все его подпараметры (параметр DCB нельзя записать с пустым значением). Заметим также, что нельзя удалять все параметры оператора DD, потому что поле операндов оператора DD не должно быть пустым (это воспринимается системой как ошибка). Если по каким-либо причинам нужно запретить действия с набором данных, то нужно добавить к описывающему его оператору DD параметр DUMMY. Это сделает набор данных фиктивным.

Модифицируемые операторы DD нужно записать в том порядке, в каком они записаны в процедуре. Вслед за модифицируемыми операторами DD одного пункта процедуры записывают все операторы DD, добавляемые к этому пункту. Добавляемые операторы DD также должны иметь формат (2.2), причем dd-имена этих операторов DD должны отличаться от всех dd-имен в пункте процедуры (при совпадении dd-имен происходит не добавление, а модификация оператора DD).

Пример 3. Изменение и добавление операторов DD в каталогизированной процедуре редактирования.

В библиотеке процедур ОС ЕС SYS1.PROCLIB имеется каталогизированная процедура редактирования LKED:

```
//LKED      EXEC PGM=IEWL,PARM='XREF,LIST,LET,NCAL',
//                REGION=96K
//SYSPRINT  DD   SYSOUT=A
//SYSLIN    DD   DDNAME=SYSIN
//SYSLMOD   DD   DSNAME=&&GOSET(GO),SPACE=(1024,(50,20,1)),
//                UNIT=SYSDA,DISP=(MOD,PASS)
//SYSUT1    DD   UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//                SPACE=(1024,(200,20))
```

Назначение наборов данных, определенных в процедуре: SYSPRINT — набор данных для листинга редактора связей; SYSLIN — набор данных, содержащий объектный модуль, подлежащий редактированию; SYSLMOD — набор данных, в который помещается загрузочный модуль, выработанный редактором; SYSUT1 — промежуточный (рабочий) набор данных редактора.

Читателю рекомендуется внимательно рассмотреть процедуру LKED. Смысл каждого параметра должен быть понятен. Если это не так, то следует повторно проработать материал, изложенный в п. 2.2.2 и разд. 2.3

Требуется, используя процедуру LKED, отредактировать объектный модуль, полученный компилятором ПЛ/1 и находящийся в каталогизированном наборе данных OBJMOD. После завершения пункта задания набор данных OBJMOD нужно уничтожить. Отредактированный загрузочный модуль нужно поместить в каталогизированную библиотеку LOADLIB под именем PAYROLL. Для включения в загрузочный модуль подпрограмм из библиотеки компилятора ПЛ/1 (ее имя SYS1.PL1LIB) к процедуре нужно добавить оператор с dd-именем SYSLIB, определяющий эту библиотеку.

Решение:

```
//LIBUPDTE        EXEC LKED
//LKED.SYSLMOD    DD DSN=LOADLIB(PAYROLL),DISP=OLD
//LKED.SYSLIB     DD DSN=SYS1.PL1LIB,DISP=SHR
//LKED.SYSIN      DD DSN=OBJMOD,DISP=(OLD,DELETE)
```

Напомним, что язык управления заданиями позволяет ссылаться на наборы данных, определенные в процедурах. Такие ссылки записывают в формате:

*.имя пункта задания.имя пункта процедуры.dd-имя

Здесь «имя пункта задания» указывает пункт, в котором вызывается процедура.

Пример 4. Выполнение загрузочного модуля, созданного при выполнении процедуры LKED в примере 3.

```
//ST EXEC PGM=*.LIBUPDTE.LKED.SYSLMOD
```

Если первый подпараметр параметра MSGLEVEL в операторе JOB не равен нулю, то операторы процедуры во входном потоке выдаются на печать, а если этот подпараметр равен единице, то печатаются операторы каталогизированных процедур. В первых двух колонках листинга для каталогизированных процедур печатаются такие знаки:

XX — если оператор не изменяется;
X/ — если некоторые параметры оператора изменены;
XX* — если оператор рассматривается системой как комментарий;
*** — если это оператор комментария.

Для процедур во входном потоке знак «X» заменяется знаком «+».

Глава 3

ТРАНСЛЯЦИЯ

3.1. ТРАНСЛЯТОРЫ

3.1.1. Характеристики

В системе программирования ОС ЕС имеются такие трансляторы: Ассемблер, компиляторы Фортрана, Кобола, ПЛ/1, Алгола-60 и транслятор для языка РПГ. Основное назначение трансляторов — перевод операторов исходного модуля на машинный язык и формирование объектного модуля. Дополнительно по указанию программиста каждый транслятор может печатать листинг, содержащий текст исходного модуля, сведения об объектном модуле и диагностические сообщения об ошибках, обнаруженных во время трансляции.

В начале трансляции Ассемблер и компиляторы Кобола и ПЛ/1 могут дополнить исходный модуль текстами на входных языках из указанных программистом библиотек. На этом этапе Ассемблер и компилятор ПЛ/1 выполняют также макрообработку, заменяя макрокоманды Ассемблера или препроцессорные утверждения ПЛ/1 порождаемыми ими текстами.

Для выполнения действий, часто повторяющихся в разных программах (распределение памяти для размещения данных, управление операциями ввода-вывода, вычисление элементарных функций и т. п.), компиляторы включают в объектный модуль обращения к стандартным библиотечным подпрограммам. Все компиляторы имеют собственные библиотеки подпрограмм, в которых каждая подпрограмма, оформленная в виде загрузочного модуля, составляет отдельный раздел. Библиотечные подпрограммы являются замкнутыми. Они объединяются с основным объектным (или загрузочным) модулем в единый загрузочный модуль на этапе редактирования.

У Ассемблера и транслятора РПГ нет библиотек стандартных подпрограмм, однако Ассемблер имеет библиотеку макроопределений SYS1.MACLIB. Она содержит тексты макроопределений для системных макрокоманд. Макроопределения, по существу, представляют собой открытые подпрограммы, которые включаются в объектный модуль во время ассемблирования.

Как известно [6], на разных этапах подготовки программы к компилятору предъявляются разные требования. Пока программа отлаживается, компилятор должен обеспечивать быструю трансляцию и возможно более полный контроль ошибок. Требования к ка-

честву объектной программы на этом этапе невысоки. При трансляции отлаженной программы, предназначенной для длительного использования, требования изменяются: теперь транслятор должен обеспечить минимальное время выполнения объектной программы, а время трансляции не имеет определяющего значения. Чтобы выполнить эти противоречивые требования, на практике используют два пути: либо создают два компилятора — отладочный и оптимизирующий, либо разрабатывают параметрический компилятор, способный работать и в отладочном и в оптимизирующем режимах.

В системе программирования ОС ЕС использованы оба пути. В частности, для Фортрана имеются два компилятора: отладочный (Фортран стандартный) и оптимизирующий (Фортран оптимизирующий), а компилятор ПЛ/1, как и Фортран оптимизирующий, построен по параметрическому принципу. Указывая для этого компилятора значение уровня оптимизации, можно управлять скоростью трансляции и качеством объектной программы.

Трансляторы ОС ЕС хранятся в общей системной библиотеке SYS1.LINKLIB в виде загрузочных модулей. Имя загрузочного модуля есть имя программы транслятора. Чтобы выполнить трансляцию, нужно указать это имя в параметре PGM оператора EXEC. Кроме такого «статического» вызова, любой транслятор можно вызвать динамически во время выполнения проблемной программы (например, из программы на языке Ассемблера). Для этого в макрокоманде динамического вызова нужно указать имя программы транслятора и dd-имена используемых транслятором наборов данных.

В табл. 3.1, составленной по материалам коллективной монографии [7] и эксплуатационной документации, приведены характеристики тех трансляторов, которые обычно поставляются в составе ОС ЕС. В колонке MFT этой таблицы показан минимальный объем памяти, требующейся для выполнения программы транслятора (напомним, что часть памяти занята ядром управляющей программы ОС), а в колонке MVT — значение параметра REGION, записанное в каталогизированных процедурах трансляции (см. п. 3.2.1). Следует иметь в виду, что при увеличении объема выделяемой транслятору основной памяти трансляция, как правило, происходит быстрее за счет уменьшения числа обращений к внешней памяти в процессе трансляции. Для компиляторов в табл. 3.1 приведены имена и объем библиотек подпрограмм, а для Ассемблера указано имя и объем библиотеки макроопределений (предполагается, что библиотека макроопределений состоит из блокированных записей; если записи несблокированы, то объем этой библиотеки возрастает почти в 1,7 раза).

Качество объектных программ, вырабатываемых трансляторами ОС ЕС, исследовалось в разных работах (см., например [9]). Отношение времени выполнения объектной программы, полученной компиляцией, ко времени выполнения программы той же задачи, полученной ассемблированием, существенно зависит от особенностей алгоритма задачи и набора использованных средств

Таблица 3.1

Характеристики трансляторов системы программирования ОС ЕС

Транслятор	Имя программы	Объем транслятора, Кбайт	Необходимая основная память, Кбайт, в режиме		Библиотеки трансляторов	
			MFT	MVT	Имя	Объем, Кбайт
Ассемблер	IEUASM	147	44	50	SYS1.MACLIB	1704
Алгол-60	ALGOL	133	44	48	SYS1.ALGLIB	49
Кобол (ANS)	IKFCBL00	430	80	86	SYS1.COBLIB	38,5
Фортран стандартный	IEYFORT	84	80	100	SYS1.FORTLIB	91
Фортран оптимизирующий	IEKA00	446	152	228	SYS1.FORTLIB	91
ПЛ/1	IEMAA	1263	44	52	SYS1.PL1LIB	276,5
РПГ	IESRPG	175	15	52	—	—

входного языка. Оно может меняться в широких пределах. В среднем по результатам нескольких экспериментов для вычислительных задач это отношение оказалось равным для Фортрана стандартного 1,8, для Фортрана оптимизирующего — 1,2, для ПЛ/1 — 1,5, для Алгола-60 — 2,7, причем в отдельных случаях для Фортрана оптимизирующего получалась единица, а для Алгола-60—16.

3.1.2. Наборы данных

Трансляторы используют входные, промежуточные и выходные наборы данных со стандартными dd-именами. Последние нужно указывать в задании на трансляцию.

Общая схема использования наборов данных при трансляции показана на рис. 3.1. Набор данных с dd-именем SYSLIB библиотечный. Все остальные наборы данных последовательные.

В табл. 3.2 приведены dd-имена наборов данных каждого транслятора (dd-имена обязательных наборов данных набраны полужирным шрифтом) и указаны групповые имена устройств, на которых они располагаются. Напомним, что стандартные групповые имена устройств имеют такие значения: SYSSQ — магнитные ленты и диски (барабаны), SYSDA — магнитные диски (барабаны), SYSCP — карточные перфораторы.

Входные наборы данных трансляторов непременно включают набор данных с dd-именем SYSIN, содержащий исходный модуль. Как правило, этот набор данных записан на перфокартах и находится во входном потоке (после оператора DD * или DATA). Иногда исходный модуль предварительно записывают на магнитную ленту или на диск.

Ассемблер и компиляторы Кобола и ПЛ/1 могут также использовать дополнительно входной библиотечный набор данных на диске с dd-именем SYSLIB. Для Ассемблера он определяет системную библиотеку SYS1.MACLIB, в которой содержатся макро-

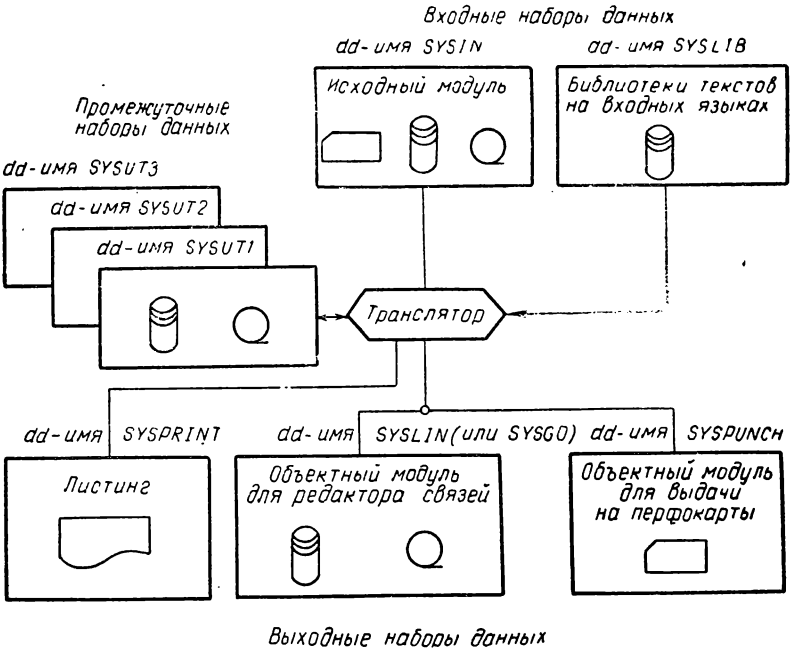


Рис. 3.1. Наборы данных трансляторов

определения системных макрокоманд и могут находиться тексты на языке Ассемблера, включаемые в программу оператором COPY. С библиотекой SYS1.MACLIB в задании иногда сцепляют личные библиотеки макроопределений и текстов на языке Ассемблера.

Для компилятора Кобол (ANS) набор данных с dd-именем SYSLIB необходим лишь в том случае, когда в исходном модуле имеются пункты COPY или утверждения INCLUDE, а для компилятора ПЛ/1 — когда в исходном модуле есть утверждения %INCLUDE, в которых не указана используемая библиотека (если библиотека указана, то набор данных с dd-именем SYSLIB не нужен).

Каждое макроопределение и каждый текст, копируемый из библиотеки, составляет один раздел библиотеки.

Промежуточные (рабочие) наборы данных на магнитных дисках или лентах используются для внутренних нужд трансляторов. Они необходимы всем трансляторам, кроме компиляторов Фортрана, которые используют большую область основной памяти (см. табл. 3.1), позволяющую в основных режимах обойтись без дополнительной внешней памяти.

Компилятор Фортран оптимизирующий использует два промежуточных набора данных на диске только в некоторых режимах работы.

Компилятору ПЛ/1 нужен промежуточный набор данных на диске с dd-именем SYSUT1 лишь в том случае, когда основной

Таблица 3.2

Наборы данных трансляторов

Транслятор	Наборы данных		
	входные	промежуточные	выходные
Ассемблер	**SYSIN** (входной поток или SYSSQ) SYSLIB (SYSDA)	**SYSUT1**, **SYSUT2**, **SYSUT3** (SYSSQ)	SYSGO (SYSSQ) SYSPUNCH (SYSCP — SYSOUT=B) **SYSPRINT** (АЦПУ — SYSOUT=A)
Алгол-60	**SYSIN** (входной поток или SYSSQ)	**SYSUT1**, **SYSUT2** (SYSSQ) **SYSUT3** (SYSDA)	SYSLIN (SYSSQ) SYSPUNCH (SYSCP — SYSOUT=B) **SYSPRINT** (АЦПУ — SYSOUT=A)
Кобол (ANS)	**SYSIN** (входной поток или SYSSQ) SYSLIB (SYSDA)	**SYSIT1** (SYSDA) **SYSUT2**, **SYSUT3**, **SYSUT4** (SYSSQ или SYSDA)	SYSPUNCH (SYSSQ или SYSCP) SYSLIN (SYSSQ) **SYSPRINT** (АЦПУ — SYSOUT=A)
Фортран	**SYSIN** (входной поток или SYSSQ)	Только Фортран оптимизирующий: SYSUT1, SYSUT2 (SYSSQ)	SYSLIN (SYSSQ) SYSPUNCH (SYSCP — SYSOUT=B) **SYSPRINT** (АЦПУ — SYSOUT=A)
ПЛ/1	**SYSIN** (входной поток или SYSSQ) SYSLIB (SYSDA)	SYSUT1 (SYSDA) SYSUT3 (SYSSQ)	SYSLIN (SYSSQ) SYSPUNCH (SYSCP — SYSOUT=B) **SYSPRINT** (АЦПУ — SYSOUT=A)
РПГ	**SYSIN** (входной поток или SYSSQ)	**SYSUT1**, **SYSUT2**, **SYSUT3** (SYSSQ)	SYSGO (SYSSQ) SYSPUNCH (SYSCP — SYSOUT=B) **SYSPRINT** (АЦПУ — SYSOUT=A)

памяти недостаточно. Однако на практике этот набор данных лучше определять всегда, считая его обязательным. Набор данных с dd-именем SYSUT3 требуется только в двух случаях: когда исходная программа на ПЛ/1 записана в 48-литерном алфавите (вместо стандартного 60-литерного алфавита) и когда в исходной программе имеются препроцессорные утверждения (операторы периода компиляции) или утверждения %INCLUDE. В этих случаях перед началом трансляции исходный модуль обрабатывается специальными препроцессорами, входящими в состав компилятора. Препроцессор 48-литерного алфавита переводит программу в стандартный 60-литерный алфавит, а препроцессор периода компиляции выполняет макрообработку. Оба препроцессора записывают переработанный текст в промежуточный набор данных с dd-именем SYSUT3.

Выходные наборы данных трансляторов включают набор данных на диске или магнитной ленте, в который помещается объектный модуль, передаваемый редактору связей, набор данных в выходном потоке для выдачи объектного модуля на перфокарты (dd-имя SYSPUNCH) и набор данных в выходном потоке для выдачи листинга (dd-имя SYSPRINT).

Объектный модуль для редактора связей компиляторы помещают в набор данных с dd-именем SYSLIN, а Ассемблер и транслятор РПГ — в набор данных с dd-именем SYSGO.

Обязательным является по крайней мере один из выходных наборов данных. Если исходный модуль транслируют с целью выявления лексических и синтаксических ошибок, то можно задать только набор данных с dd-именем SYSPRINT.

3.1.3. Опции

Трансляторы ОС ЕС предоставляют программисту ряд необязательных возможностей, которыми он может пользоваться по своему выбору для управления режимом трансляции и составом выходной информации. Эти возможности задают в операторе EXEC, вызывающем транслятор, в виде подпараметров параметра PARM и называют *опциями* (option — выбор, возможность выбора). Заметим, что список опций, как и любой список подпараметров параметра PARM, не должен превышать 100 символов, включая разделяющие запятые. Опции трансляторов задают ключевыми подпараметрами, поэтому в списке они могут располагаться в произвольной последовательности.

Различают два типа опций: задаваемые ключевыми словами и задаваемые ключевым словом и значением.

Ключевыми словами задают опции, которые имеют два (иногда три) фиксированных значения. Большинство таких опций имеет утвердительную и отрицательную формы. Например, SOURCE — печатать исходную программу, NOSOURCE — не печатать.

Ключевым словом и значением задают опции, которым можно назначить произвольное значение из некоторого множества значе-

ний. Например, опция NAME=GAUSS указывает имя программы в Фортране (NAME — ключевое слово, GAUSS — значение), а опция SIZE=56K задает объем основной памяти, доступной компилятору ПЛ/1.

Почти все опции имеют стандартное значение, которое принимается по умолчанию, если в задании опция явно не указана.

Набор опций в разных трансляторах различен (от 4 опций в трансляторе РПГ до 25 опций в компиляторе ПЛ/1). Все опции трансляторов можно разделить на пять групп:

1) опции вывода, указывающие куда направить объектный модуль: редактору или на перфорацию;

2) опции ввода, определяющие особенности кодировки исходного модуля и формат карт (в ПЛ/1);

3) опции листинга, определяющие его состав;

4) управляющие опции, задающие режим трансляции и особенности формируемой объектной программы;

5) специальные опции, специфичные для данного транслятора.

Чаще всего программисты используют опции листинга и управляющие опции. Перечень опций трансляторов с их стандартными значениями приведен в приложении 2. При генерации системы стандартные значения опций могут быть изменены, поэтому в конкретном варианте ОС ЕС стандартные значения могут отличаться от значений, указанных в приложении 2.

Пример 1. Использование опций компилятора Фортрана (см. приложение 2).

```
//EXOPT1 EXEC PGM=IEYFORT,PARM='LIST,MAP'
```

Этот оператор вызывает компилятор Фортран стандартный (имя программы IEYFORT). Объектный модуль будет помещен в набор данных для редактора (LOAD по умолчанию). В листинге будут напечатаны такие элементы: исходная программа (SOURCE по умолчанию), объектная программа (LIST) и таблица имен переменных с их адресами (MAP).

Пример 2. Использование опций компилятора ПЛ/1 (см. приложение 2).

```
//EXOPT2 EXEC PGM=IEMAA,PARM='OPT=0,SIZE=56K,N=PR1'
```

Этот оператор предписывает выполнить компиляцию программы ПЛ/1 без оптимизации объектной программы (OPT=0). Компилятору доступна память 56K, после редактирования загрузочному модулю будет назначено имя PR1. По умолчанию в листинг будет включен список действующих опций и текст исходной программы (OPLIST и SOURCE2).

Время компиляции в общем случае возрастает с уменьшением доступной компилятору памяти, определяемой опцией SIZE и увеличением уровня оптимизации (при OPT=2 время трансляции максимально). Для ПЛ/1 желательно указывать в SIZE не менее 56K, тогда небольшие программы транслируются без набора данных на диске SYSUT1.

3.2.1. Состав и особенности

Для выполнения трансляции нужно написать задание. Чтобы упростить составление задания и сократить его объем, рекомендуется использовать каталогизированные процедуры трансляции, постоянно хранящиеся в системной библиотеке процедур SYS1.PROCLIB.

Для описываемых здесь трансляторов каталогизированные процедуры трансляции имеют такие имена: ASMFC, ALGOFC, COBUC, FORTGC, FORTHC, PLILFC и RPGEC. Первые четыре или пять букв имени — мнемоническое обозначение транслятора, последняя буква — С — указывает назначение процедуры (compilation). Например, ASMFC — каталогизированная процедура ассемблирования для Ассемблера, FORTGC — каталогизированная процедура трансляции для компилятора Фортран стандартный, а FORTHC — для компилятора Фортран оптимизирующий.

Задание на трансляцию с использованием каталогизированной процедуры обычно записывают так:

```
//имя задания JOB параметры
//имя пункта EXEC имя каталогизированной процедуры
//имя пункта процедуры.SYSIN DD *
```

исходный модуль

```
/*
```

Оператор с dd-именем «имя пункта процедуры. SYSIN» обязателен, потому что тексты каталогизированных процедур не включают в себя описание входного набора данных транслятора, содержащего исходный модуль. Чтобы написать такое задание, нужно знать имя пункта процедуры, записанное в тексте процедуры. Следует ознакомиться с текстами каталогизированных процедур, потому что иногда приходится изменять в них некоторые параметры и добавлять операторы описания данных (операторы DD). Кроме того, эти тексты могут служить образцами для написания других заданий. Ниже приведены тексты каталогизированных процедур трансляции.

Каталогизированная процедура ассемблирования ASMFC:

```
//ASM     EXEC PGM=IEUASM,REGION=50K
//SYSLIB  DD   DSNAME=SYS1.MACLIB,DISP=SHR
//SYSUT1  DD   DSNAME=&SYSUT1,UNIT=SYSSQ,                    *
//             SPACE=(1700,(477,50)),SEP=(SYSLIB)
//SYSUT2  DD   DSNAME=&SYSUT2,UNIT=SYSSQ,                    *
//             SPACE=(1700,(400,50))
//SYSUT3  DD   DSNAME=&SYSUT3,SPACE=(1700,(400,50)),         *
//             UNIT=SYSSQ,SEP=(SYSUT2,SYSUT1,SYSLIB))
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
```

Каталогизированная процедура трансляции ALGOFC:

```
//ALGOL     EXEC PGM=ALGOL,REGION=48K
//SYSPRINT  DD   SYSOUT=A
//SYSPUNCH  DD   SYSOUT=B
//SYSLIN    DD   DSN=&LOADSET,UNIT=SYSSQ,SEP=SYSPUNCH,                *
//               SPACE=(3600,(10,4)),DISP=(MOD,PASS)
//SYSUT1    DD   DSN=&SYSUT1,UNIT=SYSSQ,SEP=SYSPRINT,                 *
//               SPACE=(1024,(50,10))
//SYSUT2    DD   DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,                   *
//               SPACE=(1024,(50,10))
//SYSUT3    DD   DSN=&SYSUT3,UNIT=SYSDA,
//               SPACE=(1024,(40,10))
```

Каталогизированная процедура COBUC:

```
//COB EXEC PGM=IKFCBL00,PARM='DECK,NOLOAD,SUPMAP',                   *
//         REGION=86K
//SYSPRINT   DD   SYSOUT=A
//SYSPUNCH   DD   SYSOUT=B
//SYSUT1     DD   UNIT=SYSDA,SPACE=(460, (700,100))
//SYSUT2     DD   UNIT=SYSDA,SPACE=(460, (700,100))
//SYSUT3     DD   UNIT=SYSDA,SPACE=(460, (700,100))
//SYSUT4     DD   UNIT=SYSDA,SPACE=(460, (700,100))
```

Каталогизированная процедура трансляции FORTGC:

```
//FORT       EXEC PGM=IEYFORT,REGION=100K
//SYSPRINT   DD   SYSOUT=A
//SYSPUNCH   DD   SYSOUT=B
//SYSLIN     DD   DSNAME=&LOADSET,DISP=(MOD,PASS),                    *
//               UNIT=SYSSQ,SPACE=(80, (200, 100),RLSE),             *
//                DCB=BLKSIZE=80
```

Каталогизированная процедура трансляции FORTHC:

```
//FORT       EXEC PGM=IEKA00,REGION=228K
//SYSPRINT   DD   SYSOUT=A
//SYSPUNCH   DD   SYSOUT=B
//SYSLIN     DD   DSNAME=&LOADSET,UNIT=SYSSQ,                         *
//               DISP=(MOD,PASS),SPACE=(400,(200,50),RLSE)
```

Каталогизированная процедура трансляции PL1LFC:

```
//PL1L       EXEC PGM=IEMAA,PARM='LOAD,NODECK',
//                REGION=52K                                         *
//SYSPRINT   DD   SYSOUT=A
//SYSLIN     DD   DSNAME=&&LOADSET,DISP=(MOD,PASS),                   *
//               UNIT=SYSSQ,SPACE=(80(250,100))
//SYSUT3     DD   DSNAME=&&SYSUT3,UNIT=SYSDA,                         *
//               SPACE=(80,(250,250)),DCB=BLKSIZE=80
//SYSUT1     DD   DSNAME=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,             *
//               (60,60),,CONTIG),SEP=(SYSUT3,SYSLIN),               *
//               DCB=BLKSIZE=1024
```

Каталогизированная процедура трансляции RPGEC:

```
//DEFAULT   PROC RGREGN=52K,                                          *
//               RGPARM='NODECK,LOAD,LIST,NOSEQN',                    *
//               RGU3SPC='(600,(100,20))',                            *
//               RGU2SPC='(600,(100,20))',                            *
//               RGU1SPC='(600,(100,20))',                            *
//               RGGOSPC='(80,(200,50))'
//RPG       EXEC PGM=IESRPG,REGION=&RGREGN,                           *
//               PARM=(&RGPARM)
//SYSPRINT  DD   SYSOUT=A
//SYSPUNCH  DD   SYSOUT=B
//SYSUT3    DD   DSNAME=&&SYSUT3,UNIT=SYSSQ,                          *
//               SPACE=&RGU3SPC
//SYSUT2    DD   DSNAME=&&SYSUT2,UNIT=SYSSQ,                          *
//               SPACE=&RGU2SPC
//SYSUT1    DD   DSNAME=&&SYSUT1,UNIT=SYSSQ,                          *
//               SPACE=&RGU1SPC
//SYSGO     DD   DSNAME=&&LOADSET,UNIT=(SYSSQ,                        *
//               SEP=SYSPUNCH),DISP=(MOD,PASS,DELETE),                *
//               SPACE=&RGGOSPC
```

Здесь приведены типовые процедуры. При генерации системы и в процессе ее эксплуатации их можно изменить.

Указанный в процедурах параметр REGION действует только в режиме MVT. В других режимах он игнорируется.

Отметим, что в процедуре RPGEC использованы символические параметры, о которых шла речь в п. 2.4.1. Оператор PROC определяет их стандратные значения.

Компиляторы Фортрана и ПЛ/1 допускают *пакетную компиляцию* (трансляцию нескольких исходных модулей в одном пункте задания). Каждый исходный модуль (главная программа или подпрограмма Фортрана, внешняя процедура ПЛ/1) транслируется в отдельный объектный модуль. Пакетная компиляция уменьшает затраты времени в операционной системе. Однако ошибки в одном модуле могут помешать трансляции других модулей, потому что код возврата пункта задания равен наивысшему коду возврата, полученному при обработке всех модулей. Правила пакетной компиляции описаны в следующем пункте.

Остальные трансляторы не могут работать в пакетном режиме. Для трансляции нескольких модулей здесь нужно выполнить несколько пунктов задания. Как при пакетной компиляции, так и при трансляции одного модуля в каждом пункте задания все объектные модули можно поместить в один выходной набор данных (в пакетном режиме иначе быть не может). Чтобы объектные модули в выходном наборе данных не накладывались один на другой, а набор данных передавался от пункта к пункту задания, в параметре DISP выходного набора данных (dd-имя SYSLIN или SYSGO) записывают подпараметры MOD и PASS. Именно такие значения записаны в каталогизированных процедурах. Если для размещения объектных модулей в выходном наборе данных не хватит места, то задание завершится аварийно с выдачей кода завершения D37 или E37. В этом случае нужно увеличить начальное количество памяти в параметре SPACE выходного набора данных и повторить выполнение задания.

3.2.2. Использование процедур

Каталогизированные процедуры трансляции чаще всего используют при отладке новых программ, а также при трансляции исходных модулей, написанных на разных входных языках, которые потом будут совместно редактироваться. Ниже приведены примеры использования каталогизированных процедур трансляции.

Пример 1. Ассемблирование исходного модуля без выдачи объектного модуля (печатается исходная и объектная программа).

```
//EXASS    JOB (1976—1),'ИВАНОВ И.И.',MSGLEVEL=1
//STASS    EXEC ASMFC,PARM.ASM=NODECK
//ASM.SYSIN DD *
```

исходный модуль на языке Ассемблера

```
/*
```

В операторе JOB записана учетная информация и имя программиста. Исходная и объектная программы печатаются в листинге Ассемблера, поскольку по умолчанию действует опция LIST.

Компиляторы Фортрана распознают конец одного и начало другого исходного модуля по утверждению END, заканчивающему любую программную единицу (главную программу или подпрограмму) Фортрана. Поэтому при пакетной компиляции карты всех транслируемых исходных модулей Фортрана помещают во входной набор данных компилятора подряд, ничем не разделяя модули. Каждый исходный модуль (программная единица) транслируется отдельно и переводится в отдельный объектный модуль.

Опции компилятора, указанные в операторе EXEC, вызывающем компилятор или каталогизированную процедуру трансляции, действуют при трансляции всех исходных модулей, однако опция NAME относится только к первому исходному модулю, если он главная программа. Все последующие главные программы в пакете получают имя MAIN, которое всегда назначается главной программе по умолчанию. Если первый по порядку исходный модуль в пакете подпрограмма или функция, то опция NAME игнорируется. Подпрограммы и функции получают имена, указанные в первом утверждении каждой из этих программных единиц (в утверждении SUBROUTINE или FUNCTION).

На практике пакетную компиляцию модулей Фортрана обычно применяют для трансляции одной главной программы и нескольких подпрограмм (функций) или для трансляции только подпрограмм и функций. Все полученные объектные модули записываются последовательно в один набор данных (dd-имя SYSLIN). Если указана опция DECK, то объектные модули одновременно могут выдаваться на перфокарты.

Пример 2. Пакетная компиляция модулей Фортрана с записью объектных модулей в выходной набор данных (dd-имя SYSLIN) и печатью текстов исходной и объектной программы, а также таблицы переменных.

```
//EXFORT      JOB (1976—2),'ПЕТРОВА П.П.',MSGLEVEL=1
//STFORT      EXEC FORTGC,PARM.FORT='LIST,MAP'
//FORT.SYSIN  DD *
```

исходный модуль Фортрана 1

исходный модуль Фортрана 2

```
/*
```

Если задание на этом заканчивается, то после его выполнения выходной набор данных с dd-именем SYSLIN будет уничтожен, потому что в каталогизированной процедуре FORTGC он определен как временный. В листинге будут напечатаны тексты исходных модулей (SOURCE по умолчанию) и объектных модулей (LIST), а также распределение памяти для переменных каждого модуля в относительных адресах (MAP).

Недостатком пакетной компиляции в Фортране является невозможность использования при трансляции каждого исходного модуля индивидуальных опций. В компиляторе ПЛ/1 этот недостаток устранен. Здесь внешние процедуры, входящие в пакет трансляции, разделяют утверждениями PROCESS. Каждое такое утверждение, записанное на отдельной карте, указывает компилятору начало следующего исходного модуля (внешней процедуры ПЛ/1) и используемые опции. Для первой внешней процедуры действуют опции, указанные в операторе EXEC, вызывающем компилятор или каталогизированную процедуру трансляции.

Утверждение PROCESS записывают в таком формате:

```
*PROCESS('список опций');
```

Знак «*» должен находиться в первой колонке карты. Пробелы допустимы (но не обязательны) между знаком «*» и ключевым словом PROCESS, между словом PROCESS и знаком «(», а также между знаками «(» и «'», «'» и «)», «)» и «;». Опции в списке разделяют запятыми, как в параметре PARM.

Если для всех опций используются стандартные значения, принимаемые по умолчанию, то утверждение PROCESS записывают в формате

```
*PROCESS;
```

Пример 3. Пакетная компиляция модулей ПЛ/1.

```
//EXPL1       JOB  (1976—3),'С.А.Ф.',MSGLEVEL=1
//STPAC1      EXEC PL1LFC
//PL1L.SYSIN  DD *
```

исходный модуль ПЛ/1 1

```
*PROCESS ('LOAD,NODECK');
```

исходный модуль ПЛ/1 2

```
/*
```

Здесь в утверждении PROCESS указаны те же опции, что в каталогизированной процедуре PL1LFC. По умолчанию действует опция SOURCE2, поэтому будут напечатаны тексты исходных модулей. Если требуется напечатать также тексты объектных модулей, то нужно задать опцию LIST.

Утверждение PROCESS в сочетании с опцией OBJNM позволяет при пакетной компиляции подготовить информацию для редактора связей о составе загрузочных модулей, которые должен сформировать редактор. Если при трансляции внешней процедуры указана опция OBJNM=имя (сокращенно N=имя), то компилятор помещает в конец полученного объектного модуля управляющее предложение редактора связей (см. п. 4.2.6) вида

NAME имя(R)

Здесь «имя» есть имя загрузочного модуля, указанное опцией OBJNM. В загрузочный модуль включаются все объектные модули, информация о которых находится во входном наборе данных редактора до предложения NAME.

Пример 4. Пакетная компиляция модулей ПЛ/1, которые должны быть включены редактором в два загрузочных модуля.

```
//EXPL2          JOB   (1976—4),'С.А.Ф.',MSGLEVEL=1
//STPAC2         EXEC  PL1LFC,PARM.PL1L='LOAD,NODECK,N=M1',
//PL1L.SYSIN     DD    *
```

исходный модуль A

```
*PROCESS('LOAD,NODECK,N=M2');
```

исходный модуль B

```
*PROCESS('LOAD,NODECK,N=M2');
```

исходный модуль C

```
/*
. . . . . .
```

После выполнения пункта задания STPAC2 в наборе данных, передаваемом редактору (dd-имя SYSLIN), будут записаны следующие объектные модули и предложения NAME:

Объектный модуль A
NAME M1(R)
Объектный модуль B
Объектный модуль C
NAME M2(R)

Если следующим будет выполняться пункт редактирования, то объектный модуль A войдет в загрузочный модуль M1, а объектные модули B и C будут включены в загрузочный модуль M2.

При трансляции модулей, написанных на разных входных языках системы программирования, нужно выполнить несколько пунктов задания.

Пример 5. Трансляция модуля ПЛ/1 и модуля Ассемблера с записью объектных модулей в один выходной набор данных.

```
//EXPLASS        JOB (1976—5),'СИДОРОВ С.С.',MSGLEVEL=1
//STPL           EXEC PL1LFC
//PL1L.SYSIN     DD   *
```

исходный модуль ПЛ/1

```
/*
```

```
//STASS          EXEC  ASMFC,PARM.ASM='LOAD,NODECK'
//ASM.SYSGO      DD    DSNAME=*.STPL.PL1L.SYSLIN,
//                     DISP=(MOD,PASS)
//ASM.SYSIN      DD    *
```

исходный модуль на языке Ассемблера

```
/*
. . . . . .
```

В следующем пункте задания объектные модули, полученные в пунктах STPL и STASS, могут редактироваться совместно.

Глава

РЕДАКТИРОВАНИЕ, ЗАГРУЗКА И ВЫПОЛНЕНИЕ

4.1. ОБЪЕКТНЫЙ МОДУЛЬ

4.1.1. Структура

Объектный модуль, вырабатываемый трансляторами ОС ЕС, содержит текст программы на языке машины, дополненный служебной информацией, и управляющие словари.

Текст программы в объектном модуле состоит из одной или нескольких программных секций. *Программная секция* — это наименьшая независимо перемещаемая часть программы, которая загружается как единое целое в выделенный ей сплошной участок основной памяти. В программную секцию, в частности, преобразуются программная единица Фортрана, внешняя процедура ПЛ/1 и программная секция, написанная на языке Ассемблера. Соответствие между языковыми понятиями и понятиями объектного модуля показано в табл. 4.1. Кроме программных секций, объектный модуль может содержать сведения об *общих секциях,* соответствующих общему блоку COMMON Фортрана или общей области COM в языке Ассемблера.

Псевдорегистр — это поле основной памяти, которое не входит в объектный модуль. Оно выделяется во время выполнения загрузочного модуля и содержит указатель динамически выделяемой памяти. Понятие «псевдорегистр» используется в компиляторе ПЛ/1. В языке Ассемблера псевдорегистру соответствует внешняя фиктивная секция.

Слабая внешняя ссылка — это внешняя ссылка в языке Ассемблера, которую редактору связей запрещено обрабатывать путем автоматического вызова библиотеки. Редактор связей не разрешает слабую внешнюю ссылку, если не существует обычной внешней ссылки к тому же внешнему имени. Использование в ассемблерных программах слабых внешних ссылок позволяет при разрешении внешних ссылок путем автоматического вызова библиотечных модулей запретить вызов и включение в загрузочный модуль тех модулей, которые не нужны в данном конкретном применении. Это сокращает размер загрузочного модуля.

Связь между программными секциями в разных модулях осуществляется при помощи *внешних символов,* которые делятся на внешние имена и внешние ссылки.

Внешнее имя — это имя секции или имя дополнительной точки входа, определенной в программной секции.

Внешняя ссылка — ссылка на имя, являющееся внешним в дру-

Таблица 4.1

Соответствие понятий языков программирования понятиям объектного модуля

Объектный модуль	Ассемблер	ПЛ/1	Фортран	Кобол	Алгол-60
Программная секция	Программная секция	Главная процедура Внешняя процедура Внешняя статическая область Внутренняя статическая область	Главная программа Подпрограмма Подпрограмма-функция	Программа	Процедура
Непоименованная программная секция	Непоименованная программная секция	—	Блок данных	—	Программа
Общая секция	—	Неинициализированная внешняя область Неинициализированная внешняя статическая область	Поименованный общий блок	—	—
Непоименованная общая секция	Общая область COM	—	Непоименованный общий блок	—	—
Внешняя фиктивная секция или псевдорегистр	Внешняя фиктивная секция	Псевдорегистр	—	—	—
Дополнительная точка входа	Имя в операторе (предложении) ENTRY				
Внешняя ссылка	Имя в предложении EXTRN A — константа V — константа	Имя с явным или умалчиваемым атрибутом EXTERNAL	Имя в операторе EXTERNAL Имя в операторе CALL Имя в указателе функции	Имя в предложении CALL	—
Слабая внешняя ссылка	Слабая внешняя ссылка	—	—	—	—

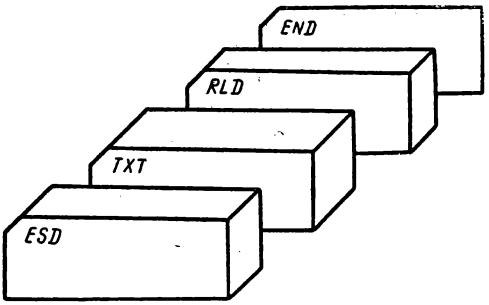


Рис. 4.1. Структура объектного модуля

гом объектном модуле. Внешние ссылки в тексте программы на машинном языке представлены *адресными константами* (константами, значениями которых являются адреса основной памяти). Значения адресных констант зависят от расположения программных секций в памяти. Поэтому эти адресные константы называют *перемещаемыми*. При помощи перемещаемых адресных констант можно также ссылаться на имена, определенные в том же объектном модуле.

Перемещаемые адресные константы корректируются дважды: при объединении объектных модулей в загрузочный модуль (эту процедуру называют *разрешением внешних ссылок*) и при загрузке загрузочного модуля в основную память для выполнения. Первую корректировку выполняет редактор связей, а вторую — программа выборки.

Информация о внешних символах и перемещаемых адресных константах находится в двух управляющих словарях: словаре внешних символов ESD (External Symbol Dictionary) и словаре перемещений RLD (Relocation Dictionary). *Словарь внешних символов* содержит все внешние имена и внешние ссылки, а *словарь перемещений* — сведения о всех перемещаемых адресных константах объектного модуля.

На любом носителе (перфокарты, магнитная лента, диск) объектные модули записываются в виде образов перфокарт. Объектный модуль содержит карты четырех типов: карты ESD (словарь внешних символов), карты TXT (текст программы), карты RLD (словарь перемещений) и карту END (признак конца объектного модуля). Карты ESD, TXT и RLD могут быть сгруппированы иначе, чем показано на рис. 4.1.

В первой колонке каждой карты записан шестнадцатеричный код 02 — признак карты объектного модуля. Колонки 2—4 содержат признак типа карты — коды символов ESD, TXT, RLD или END. Основная информация на картах ESD, TXT и RLD находится в колонках 17—72. В этих колонках записаны элементы ESD или элементы RLD, или текст программы. Колонки 73—76 могут использоваться для идентификации конкретного модуля, а колонки 77—80 содержат порядковый номер карты. В остальных колонках находится служебная информация, зависящая от типа карты, или находятся пробелы.

4.1.2. Словарь внешних символов

Словарь внешних символов содержит внешние имена и внешние ссылки (табл. 4.2) всех программных секций объектного модуля.

Таблица 4.2

Типы внешних символов

Тип	Внешний символ
SD	Имя программной секции
LD	Имя дополнительной точки входа
ER	Внешняя ссылка
CM	Имя общей секции или идентификация непоименованной общей секции
PC	Идентификация непоименованной программной секции
XD	Имя псевдорегистра или внешней фиктивной секции
WX	Слабая внешняя ссылка

В табл. 4.3 приведена информация, содержащаяся в словаре внешних символов для каждого внешнего символа.

Таблица 4.3

Информация, содержащаяся в словаре ESD

Тип	Код	Имя	ESDID	Адрес	Длина	Выравнивание	SD ESDID
SD	00	+	+	+	+	—	—
LD	01	+	—	+	—	—	+
ER	02	+	+	—	—	—	—
PC	04	—	+	+	+	—	—
CM	05	+ —	+	+	+	—	—
XD	06	+	+	—	+	+	—
WX	0A	+	+	—	—	—	—

Примечание. Наличие информации отмечено знаком «+», а отсутствие — знаком «—». Например, непоименованная программная секция PC не имеет имени, а общая секция может быть поименованной или непоименованной, поэтому для типа PC в графе «имя» записан знак «—», а для типа CM — знаки «+» и «—».

Графа ESDID содержит информацию о наличии идентификационного (порядкового) номера в словаре внешних символов. Этот номер используют для ссылок на внешние символы и для идентификации непоименованных программных секций и общих секций. Точки входа LD не имеют идентификационных номеров.

В графе «Адрес» для внешних символов типа SD, PC и CM указан начальный адрес секции, а для символов типа LD — адрес внутри программной секции.

В графе «Длина» приводится количество байт памяти, назначенное транслятором для секции.

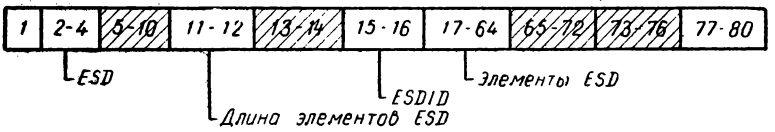


Рис. 4.2. Формат карт ESD

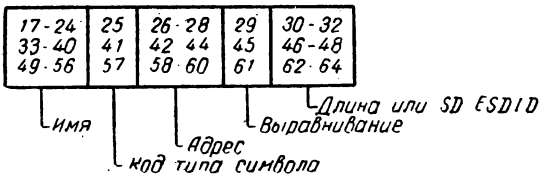


Рис. 4.3. Формат элемента ESD

«Выравнивание», указываемое только для символов типа XD, есть число, на единицу меньшее количества байт в единице выравнивания. Например, 3 означает выравнивание на границу слова (4 байта).

Для символа типа LD всегда указывается идентификационный номер имени программной секции SD ESDID, в которой этот символ определен.

При рассмотрении формата элемента ESD (рис. 4.2 и 4.3) следует иметь в виду информацию, приведенную в табл. 4.3. Элемент ESD занимает 16 байт, следовательно, в информационном поле (колонки 17—72) можно разместить до трех элементов ESD. В колонках 11—12 указывается длина (в байтах) всех элементов ESD на данной карте, а в колонках 15—16 записан ESDID первого отличного от LD символа ESD.

4.1.3. Текст

Карты текста программы TXT (рис. 4.4) содержат объектную программу и служебную информацию. Текст программы в объектном коде записан на карте с колонки 17. Каждая секция начинается с новой карты.

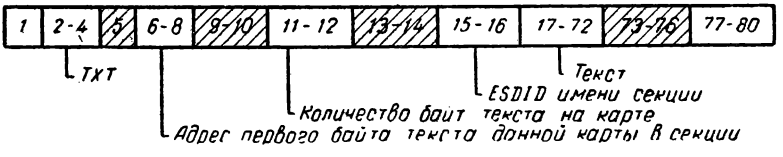


Рис. 4.4. Формат карт TXT

4.1.4. Словарь перемещений

Словарь перемещений RLD содержит сведения о всех перемещаемых адресных константах. Каждый элемент RLD хранит информацию об одной адресной константе (рис. 4.5).

Рис. 4.5. Формат карт RLD

ESDID расположения (рис. 4.6) указывает программную секцию, в которой расположена адресная константа, а ESDID перемещения указывает либо на внешнюю ссылку (ER или WX), либо на имя программной секции (SD или PC), к которой относится точка, определяемая адресной константой. Значение адресной константы зависит от адреса, который будет назначен точке, указываемой ESDID перемещения. Колонки 22—24 содержат адрес расположения адресной константы в программной секции, ESDID которой записан в колонках 19—20.

В байте флагов (рис. 4.7) биты 2—3 содержат код типа адресной константы: 00 — константа типа A или Q, 01 — константа типа V. Длина адресной константы кодируется битами 4—5: 00 — один байт, 01 — два байта, 10 — три байта, 11 — четыре байта. Бит 6 равен 0, если перемещение положительно, и 1, если оно отрицательно.

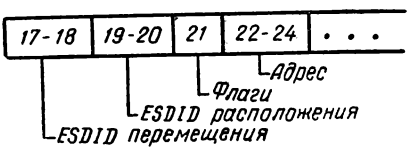


Рис. 4.6. Формат элемента RLD

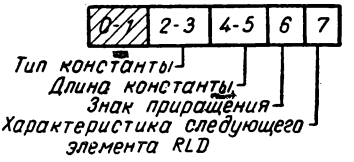


Рис. 4.7. Формат байта флагов

Каждый элемент RLD занимает либо восемь колонок карты, либо четыре. Если последний бит байта флагов равен 1, то в следующем элементе RLD не указываются ESDID расположения и перемещения. В этом случае элемент RLD занимает только четыре колонки, причем на него распространяются предшествующие ESDID расположения и перемещения.

Объектный модуль заканчивается картой END. Если точка входа в программу не указана, то соответствующие колонки на рис. 4.8 содержат пробелы. Свободные колонки карты END могут содержать сведения о версии ОС ЕС и дате получения объектного модуля.

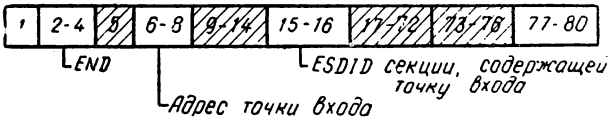


Рис. 4.8. Формат карты END

4.1.5. Пример

На рис. 4.9 и 4.10 приведены фрагменты листингов Ассемблера для двух модулей. Первый модуль состоит из непоименованной программной секции, программной секции B и общей области. Этот модуль содержит ссылку на символ E второго модуля, со-

```
                                                EXTERNAL SYMBOL DICTIONARY
SYMBOL    TYPE ID   ADDR    LENGTH LD ID

          PC   01 000000  000020
B         SD   02 000020  000014
          CM   03 000000  000004
E         ER   04

  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT

000000                              1          START
000000 0550                         2          BALR  5,0
000002                              3          USING *,5
000002 5870 500E              00010 4          L     7,=A(X)
000006 5830 5012              00014 5          L     3,=A(C)
00000A 07F3                         6          BR    3
000020                              7 B        CSECT
000020                              8          USING X-,7
000020 41180000                     9 X        DC    E'1.5'
000024 5860 5016              00018 10 C       L     6,=A(XCOM)
000028 D203 6000 7000 00000 00020 11           MVC   0(4,6),X
00002E 58F0 501A              0001C 12         L     15,=V(E)
000032 07FF                         13         BR    15
000000                              14         COM
000000                              15 XCOM    DS    E
                                    16         END
000010 00000020                     17                =A(X)
000014 00000024                     18                =A(C)
000018 00000000                     19                =A(XCOM)
00001C 00000000                     20                =V(E)
                                                RELOCATION DICTIONARY

 POS.ID   REL.ID   FLAGS   ADDRESS

   01       02      0C     000010
   01       02      0C     000014
   01       03      0C     000018
   01       04      1C     00001C
```

Рис. 4.9. Фрагмент листинга Ассемблера 1

```
                                                EXTERNAL SYMBOL DICTIONARY
BOL    TYPE ID   ADDR   LENGTH LD ID

       SD   01 000000 000006
       LD      000000          01
       CM   02 000000 000004


LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT

0000                               1 D        START
0000                               2          USING *,15
                                   3          ENTRY E
                                   4 E        RETURN RC=0
0000                               5+E        DS     0H
0000 41F0 0000                00000  6+         LA     15,0(0,0) LOAD RETURN COD
0004 07FE                          7+         BR     14 RETURN
0000                               8          COM
0000                               9 M        DS     E
                                  10          END
```

Рис. 4.10. Фрагмент листинга Ассемблера 2

стоящего из программной секции D и общей области. В первом модуле имеются также ссылки на внутренние имена X, C и XCOM.

Для обоих модулей на рисунках приведены тексты исходной (SOURCE STATEMENT) и объектной (OBJECT CODE) программ, а также словари внешних символов (EXTERNAL SYMBOL DICTIONARY) и перемещений (RELOCATION DICTIONARY). Второй модуль не содержит перемещаемых адресных констант, поэтому словарь перемещений для него пуст.

4.2. РЕДАКТОР СВЯЗЕЙ

4.2.1. Функции

Редактор связей предназначен для создания загрузочных модулей. Основная его функция — объединить указанные в задании объектные и загрузочные модули в загрузочный модуль заданной структуры. Созданный модуль редактор связей помещает в библиотеку, описанную в задании.

Если нет других указаний, то редактор связей формирует загрузочный модуль простой структуры. Для исполнения такой модуль загружается в основную память целиком. Объединяемые модули имеют собственные начальные адреса, назначенные при трансляции, и состоят из одной или нескольких программных секций. При создании загрузочного модуля простой структуры редактор связей назначает первому байту первой обрабатываемой программной секции нулевой адрес. Остальные программные секции получают адреса в порядке их поступления на обработку относительно этого начального адреса, поэтому в выходном загрузочном модуле

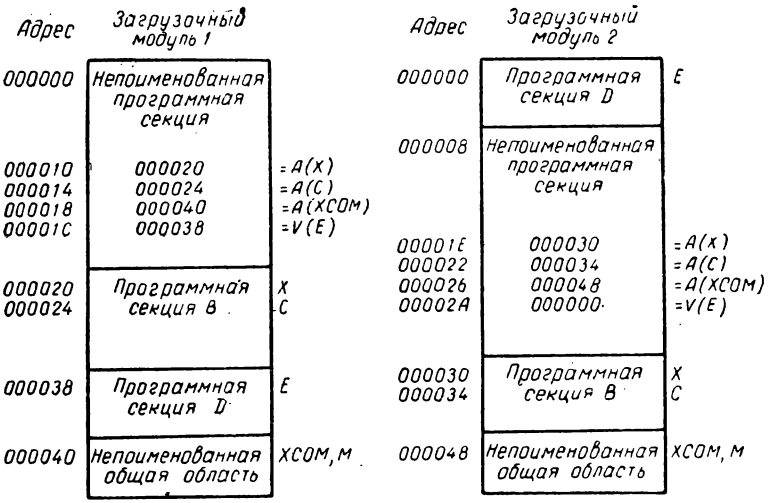


Рис. 4.11. Варианты текста загрузочного модуля

все программные секции имеют единую адресацию, соответствующую их положению в модуле. Формируя загрузочный модуль, редактор связей объединяет одноименные общие области (общие секции), определенные в разных модулях. Адрес первого байта каждой программной секции и каждой общей секции выравнивается на границу двойного слова.

Редактор связей объединяет также управляющие словари всех модулей. Объединенный словарь внешних символов называют CESD (Common External Symbol Dictionary), а объединенный словарь перемещений имеет прежнее название RLD.

В ходе объединения модулей редактор связей, используя словари внешних символов и перемещений, разрешает все межмодульные ссылки, заданные перемещаемыми адресными константами. Пусть, например, объединяются два объектных модуля, листинги которых показаны на рис. 4.9 и 4.10. На рис. 4.11 представлены два варианта структуры текста загрузочного модуля. Загрузочный модуль 1 получается, если объектные модули поступают на обработку в том порядке, в котором они приведены в листинге (рис. 4.9 и 4.10), а загрузочный модуль 2 будет получен, если порядок поступления модулей обратный. Для обоих вариантов обработки на рис. 4.11 приведены скорректированные значения адресных констант (адресные константы =A(X), =A(C) и =A(XCOM) задают внутренние ссылки, а =V(E) — внешнюю).

Схема разрешения внешних ссылок такова. В соответствии с порядком размещения программных секций, их длиной и исходными начальными адресами (длины и начальные адреса программных секций содержатся в словаре ESD) определяется перемещение каждой программной секции и каждой общей секции.

К значениям адресных констант, которые задают внутренние ссылки, добавляются перемещения секций, указанных ESDID перемещения в словаре перемещений (см. рис. 4.6).

Для разрешения внешних ссылок по имени каждой внешней ссылки редактор связей отыскивает среди внешних имен других модулей соответствующее внешнее имя. Затем значение адресной константы, которая задает внешнюю ссылку, алгебраически складывается с суммой значения внешнего имени, указанного ESDID перемещения в словаре перемещений, и перемещения секции, в которой это имя определено. Знак приращения задает байт флагов (см. рис. 4.7).

Помимо описанной основной функции — создания загрузочного модуля — редактор связей по указанию программиста может выполнять ряд дополнительных функций. Важнейшими из них являются: автоматическое разрешение внешних ссылок, оставшихся неразрешенными к концу обработки заданных модулей, путем поиска недостающих модулей в библиотеках; модификация существующих загрузочных модулей (замена и переименование программных секций); создание загрузочных модулей оверлейной структуры (структуры с перекрытием), позволяющих при выполнении программы обойтись меньшим объемом основной памяти за счет сегментации программы; создание в одном пункте задания нескольких загрузочных модулей.

Для управления выполнением этих и некоторых других функций программист может воспользоваться опциями и управляющими предложениями редактора.

4.2.2. Загрузочный модуль

Загрузочный модуль, формируемый редактором связей, структурно подобен объектному модулю и содержит ту же информацию: текст программы, объединенный словарь внешних символов и словарь перемещений. Эта информация обеспечивает настройку перемещаемых адресных констант по месту при загрузке модуля в память для выполнения, а также позволяет повторно редактировать загрузочный модуль при объединении его с другими загрузочными или объектными модулями. Однако в отличие от объектного модуля загрузочный модуль состоит не из образов перфокарт, а из записей переменной длины, в которых элементы текста программы, объединенного словаря внешних символов и словаря перемещений упакованы более плотно, чем в объектном модуле. Поэтому загрузочный модуль занимает на запоминающих устройствах меньше места, чем объектный модуль, содержащий тот же текст.

Текст программы в загрузочном модуле представляет собой совокупность программных секций совместно отредактированных модулей, настроенных на область основной памяти, начинающуюся с нулевого байта, причем начало каждой секции выравнено на границу двойного слова. На рис. 4.12 показан план текста про-

```
CONTROL SECTION                          ENTRY
  NAME      ORIGIN   LENGTH              NAME     LOCATION
*PRIVATE      00       20
 B            20       14
 D            38        6
                                         E          38
*BLANKCOM     40        4
```

Рис. 4.12. План загрузочного модуля

граммы для загрузочного модуля, представленного на рис. 4.11. В таком виде план печатается в листинге редактора связей. Служебные имена ♉ PRIVATE и ♉ BLANKCOM назначены редактором связей для непоименованной программной секции и непоименованной общей области соответственно.

4.2.3. Версии

По сложности программа редактора связей почти не уступает трансляторам. Существуют разные уровни и версии редактора связей. Версии отличаются друг от друга требованиями к объему основной памяти и быстродействием. Программы редактора связей, использующие больше основной памяти, исполняются быстрее за счет уменьшения числа обращений к внешней памяти.

Все версии программы редактора имеют оверлейную структуру и выполняются под управлением оверлейного супервизора (см. разд. 5.2). В зависимости от версии программа редактора связей при хранении на диске занимает область от 32 до 64К.

Таблица 4.3

Минимальные требования к памяти программ редактора связей, Кбайт

Версия	Имя программы	Режимы	
		PCP и MFT	MVT
F44	IEWLF440	44	54
F88	IEWLF880	88	96
F128	IEWLF128	128	136

Обычно программу редактора связей вызывают по имени IEWL. По стандартному соглашению это имя соответствует максимальной версии редактора связей, имеющейся в системе. Как и программы трансляторов, программу редактора связей можно вызвать либо «статически», указав ее имя (IEWL) в параметре PGM оператора EXEC, либо динамически из проблемной программы при помощи макрокоманд динамического вызова.

Стандартные dd-имена наборов данных редактора связей, приведенные на рис. 4.13, должны быть указаны в задании на редактирование. Обязательны только четыре набора данных (на рис. 4.13 они обведены двойной линией): набор данных основного ввода, промежуточный набор данных и оба выходных набора данных.

Набор данных основного ввода (dd-имя SYSLIN) содержит либо только объектные модули, либо управляющие предложения редактора, либо и то и другое. Этот набор данных может быть последовательным или библиотечным и может располагаться на любом устройстве (читающее устройство перфокарт, магнитная лента, диск, барабан). Если набор данных библиотечный, то в параметре DSNAME оператора DD нужно указать имя библиотеки и имя раздела. Загрузочные модули в наборе данных основного ввода находиться не могут.

Библиотеки автовызова (dd-имя SYSLIB) должны быть описаны в задании для редактора связей только в том случае, когда предусмотрено автоматическое разрешение внешних ссылок, оставшихся неразрешенными к концу формирования загрузочного модуля. Если программист предусмотрел такую возможность, то редактор связей просматривает оглавления указанных ему библиотек автовызова, отыскивает в оглавлениях имена, совпадающие с неразрешенными внешними ссылками, присоединя-

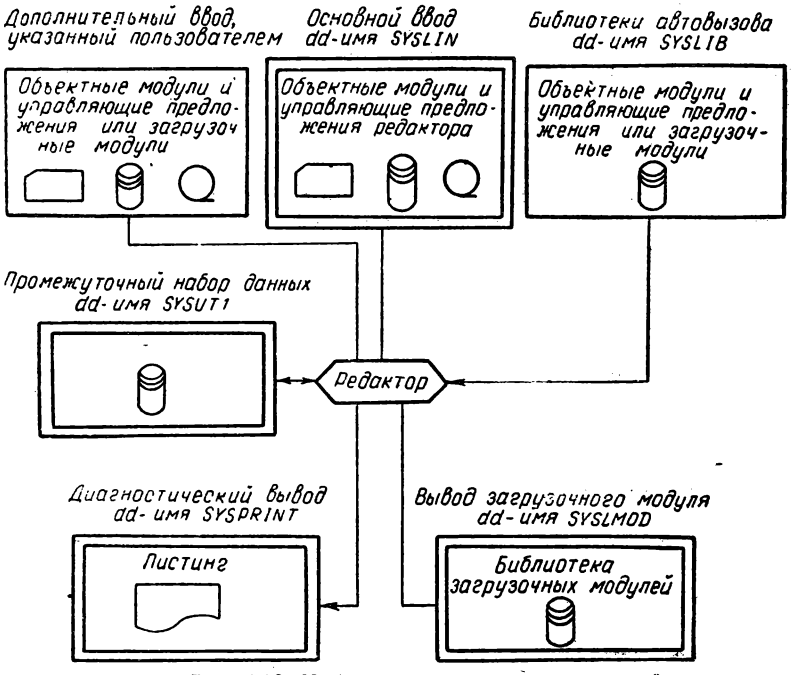


Рис. 4.13. Наборы данных редактора связей

ет модули, в которых эти имена определены как внешние, к формируемому загрузочному модулю и разрешает внешние ссылки.

Обычно библиотеки автовызова требуются, когда редактор связей обрабатывает объектные модули, полученные компиляторами Алгола-60, Фортрана, Кобола и ПЛ/1. В задании может быть указана одна библиотека или несколько сцепленных библиотек, причем в параметре DSNAME операторов DD нужно указать только имена библиотек (имена разделов не указываются).

Библиотеки автовызова находятся на дисках и могут содержать либо объектные модули, либо объектные модули и управляющие предложения редактора, либо только загрузочные модули. Смешивать в этом наборе данных загрузочные модули с объектными модулями и управляющими предложениями нельзя. Это ограничение объясняется различием формата записей, из которых состоит загрузочный модуль и объектный модуль (см. п. 4.2.2).

Наборы данных дополнительного ввода можно задать редактору при помощи управляющего предложения редактора INCLUDE (включить), описанного в п. 4.2.6. Эти наборы данных могут иметь произвольные dd-имена. Они могут находиться на любом устройстве и содержать либо объектные модули, либо объектные модули и управляющие предложения, либо только загрузочные модули. Для библиотечных наборов данных в параметре DSNAME оператора DD указывают только имена библиотек (без имен разделов).

Промежуточный набор данных (dd-имя SYSUT1) должен быть последовательным. Он всегда располагается на устройстве прямого доступа (обычно на диске) и используется для внутренних нужд редактора.

Диагностический выходной набор данных (dd-имя SYSPRINT) является последовательным. Он предназначен для листинга редактора.

Выходная библиотека загрузочных модулей (dd-имя SYSLMOD) всегда создается на диске. В нее записывают загрузочные модули, созданные редактором. Эта библиотека может быть постоянной или временной. В параметре DSNAME оператора DD нужно указать имя библиотеки. Имя раздела можно не указывать. Имя раздела, являющееся именем загрузочного модуля, можно указать в управляющем предложении редактора связей NAME, описанном в п. 4.2.6. Если имя раздела не назначено ни в операторе DD, ни в предложении NAME, то редактор связей назначит временное имя.

4.2.5. Опции

Все опции редактора связей, кроме опции, указывающей размер области памяти, доступной редактору, задают ключевыми словами. Для этих опций в отличие от опций трансляторов явно существует только одна форма: утвердительная или отрицательная. Противоположная форма принимается по умолчанию.

Редактор связей имеет 16 опций. Их можно разделить на две группы: управляющие опции и опции листинга.

Ниже описаны часто используемые опции и некоторые опции, характеризующие возможности редактора связей. Полный перечень опций приведен в приложении 2.

Управляющие опции приписывают модулю атрибуты, которые записываются в оглавление библиотеки загрузочных модулей вместе с именем модуля, определяют особенности структуры модуля и задают режим редактирования.

NE приписывает модулю атрибут НЕРЕДАКТИРУЕМЫЙ. Такой модуль не имеет словаря внешних символов и занимает поэтому меньше места в библиотеке. Его нельзя повторно редактировать. Нередактируемый модуль предназначен только для выполнения.

OVLY приписывает модулю атрибут ОВЕРЛЕЙ и требует, чтобы редактор создал модуль оверлейной структуры (см. разд. 5.2).

SCTR приписывает модулю атрибут ФОРМАТ ВРАЗБРОС и требует, чтобы редактор связей сформировал модуль, пригодный для загрузки и выполнения в ряде несмежных участков памяти. Это позволяет повысить эффективность использования свободной основной памяти. Если эта возможность не задана, то загрузочный модуль можно выполнить только в сплошной области памяти.

LET приписывает модулю атрибут ВЫПОЛНИМЫЙ даже в том случае, если во время редактирования обнаружены ошибки серьезности 2 (код возврата 8), которые могут сделать выполнение модуля невозможным (например, не все внешние ссылки разрешены). Эту опцию можно использовать при автономной отладке модулей, которые в дальнейшем будут комплексироваться. Если неразрешенные внешние ссылки фактически не используются, то выполнение загрузочного модуля произойдет нормально. В случае оверлейной структуры опцию LET заменяют опцией XCAL, позволяющей игнорировать неправильные межсегментные ссылки (см. разд. 5.2).

NCAL запрещает редактору связей использовать аппарат автоматического вызова библиотек. Модулю приписывается атрибут ВЫПОЛНИМЫЙ даже при неразрешенных внешних ссылках. Эту опцию можно использовать для экономии места в библиотеке загрузочных модулей, поскольку формируемый загрузочный модуль не будет содержать библиотечных подпрограмм. Перед исполнением такой модуль нужно редактировать повторно. Опция NCAL позволяет экономно использовать память на дисках, когда программа каждый раз собирается из независимо транслированных модулей в разных сочетаниях, как это бывает при использовании пакетов прикладных программ. Библиотеки пакетов прикладных программ выгоднее составлять из загрузочных модулей, а не из объектных, потому что загрузочные модули за счет более плотной упаковки записей занимают меньше места.

SIZE=(V1,V2) определяет размер памяти, доступный редактору связей—V1, и размер буфера загрузочного модуля — V2. Значения V1 и V2 задают числом байт или килобайт (табл. 4.4.). В любом случае значение V1 должно быть больше V2 и не должно превосходить объема основной памяти. Опция SIZE позволяет увеличить быстродействие редактора связей. При достаточно большой доступной памяти редактор связей может обойтись без использования промежуточного набора данных на диске (dd-имя SYSUT1). Если эта опция не задана или задана неверно, то принимается ее значение, установленное при генерации системы.

Таблица 4.4

Значение V1 и V2 (Кбайт) опции SIZE и соотношение между ними

Версия редактора	Минимальная величина V1	Величина V2		Минимальная величина разности V1—V2
		минимальная	максимальная	
F44	44	6	100	38
F88	88	6	100	44
F128	128	6	100	66

Опции листинга определяют состав листинга. Этих опций три: LIST — предписывает напечатать список управляющих предложений, MAP — напечатать план выходного модуля, XREF — напечатать таблицу перекрестных ссылок (для внешних ссылок) и план выходного модуля.

Некоторые опции редактора связей несовместимы. В частности, несовместимы опции OVLY и SCTR, XREF и MAP, XREF и NE, MAP и NE. Другие опции, наоборот, должны использоваться совместно. Например, опция XCAL применима только совместно с OVLY.

Пример. Требуется выполнить редактирование.

Доступная память — 50K. Нужно напечатать список управляющих предложений редактора и план модуля.

Решение:

```
//EXLE1 EXEC PGM=IEWL,PARM = 'SIZE=(50K,6K),LIST,MAP'
```

4.2.6. Управляющие предложения

Управляющие предложения, как и опции, служат для управления работой редактора. Однако в отличие от опций, которые записывают в параметре PARM оператора EXEC, управляющие предложения находятся во входных наборах данных редактора.

При записи на стандартном бланке каждое управляющее предложение редактора записывают начиная с новой строки. Это соответствует новой перфокарте. Первая позиция бланка (первая колонка перфокарты) должна быть пустой. За ней следуют два поля: поле операции и поле операндов, разделенные пробелами.

Внутри полей пробелы недопустимы. Перед полем операции может быть больше одного пробела.

В поле операции записывают название управляющего предложения.

Поле операндов содержит один операнд или несколько операндов, разделенных запятыми. На карте поле операндов записывают не далее колонки 71. Однако его можно продолжить на одной или нескольких картах продолжения. Прерванное поле операндов должно заканчиваться запятой, записанной после операнда, а в колонке 72 должен быть записан любой знак, отличный от пробела. На следующей карте поле операндов продолжают начиная с колонки 16.

Редактор имеет одиннадцать управляющих предложений. Ниже описаны те из них, которые программисты обычно используют при создании загрузочных модулей простой структуры. Управляющие предложения для модулей оверлейной структуры описаны в разд. 5.2.

Управляющее предложение

NAME имя модуля

указывает имя загрузочного модуля, формируемого редактором. Это предложение может находиться только в наборе данных основного ввода (dd-имя SYSLIN). Оно помещается после всех объектных модулей и управляющих предложений, которые используются при создании данного загрузочного модуля. Имя модуля должно содержать не более восьми букв и цифр и должно начинаться с буквы.

Если в наборе данных основного ввода находится несколько предложений NAME, то каждое из них заканчивает информацию, используемую для создания одного загрузочного модуля. Это позволяет формировать в одном пункте задания несколько загрузочных модулей, т. е. выполнять *пакетное редактирование.*

Если вновь созданный редактором модуль должен заменить в библиотеке загрузочных модулей старый модуль с тем же именем, то предложение NAME записывают в форме

NAME имя модуля(R)

Пример 1. Пакетное редактирование и замена загрузочных модулей в библиотеке новыми.

```
//SYSLMOD   DD  DSNAME=ALIB,DISP=(MOD,KEEP),...
//SYSLIN    DD  DSNAME=&&LOADSET,DISP=OLD,...
//          DD  *
              NAME   MOD5(R)
```

объектный модуль A

```
              NAME   MOD6(R)
/*
```

Операторы DD и управляющие предложения обеспечивают создание двух загрузочных модулей. Модуль MOD5 объединяет все объектные модули, имеющиеся в наборе данных &&LOADSET, а модуль MOD6 создается на основе объектного модуля A, находящегося во входном потоке. Оба загрузочных моду-

ля записываются в библиотеку ALIB. Если в этой библиотеке уже есть модули с именами MOD5 и MOD6, то они будут заменены вновь созданными модулями с теми же именами.

В отличие от предложения NAME все остальные управляющие предложения редактора связей могут находиться в любом входном наборе данных, а не только в наборе данных основного ввода.

Управляющее предложение

ENTRY внешнее имя

указывает имя основной точки входа модуля. Это имя нужно задавать всегда, когда загрузочный модуль вновь редактируется, а также в тех случаях, когда точка входа не совпадает с первым байтом программы.

Управляющее предложение

ALIAS список имен

указывает дополнительные имена модуля и дополнительные точки входа. Имя в списке имен предложения ALIAS считается дополнительной точкой входа, если оно определено в этом модуле как внешнее имя, в противном случае оно рассматривается как дополнительное имя модуля (псевдоним).

Дополнительные имена и дополнительные точки входа записываются в оглавление библиотеки загрузочных модулей вместе с основным именем модуля. Дополнительные имена и дополнительные точки входа называют *алиасами*. Модуль может иметь до 16 алиасов. Алиасы можно задать в одном или нескольких предложениях ALIAS. Заметим, что это предложение не уничтожает в оглавлении ранее записанных алиасов.

Загрузочный модуль можно вызвать по основному имени или по любому алиасу. Если при вызове указано основное или дополнительное имя модуля, то его выполнение начнется с основной точки входа, однако если при вызове названа дополнительная точка входа, то выполнение модуля начнется с этой точки.

Пример 2. Назначение загрузочному модулю имени, точки входа и алиасов.

```
//SYSLIN   DD *
  ENTRY    BEGIN
  ALIAS SIM,DUALSIM,SIPR
  NAME SIPLEX
/*
```

Управляющее предложение

INCLUDE список наборов данных

указывает наборы данных дополнительного ввода. Последовательные наборы данных задают их dd-именами, а для библиотечных наборов данных нужно записать dd-имена и имена разделов (или их алиасы) в виде

dd-имя(список имен разделов)

Каждому dd-имени, упомянутому в предложении INCLUDE, в пункте задания, где выполняется редактирование, должен соответствовать оператор DD.

Пример 3. Указание наборов данных дополнительного ввода для создания загрузочных модулей LMOD1 и LMOD2.

```
//SEQSET   DD   DSNAME=SET,DISP=OLD,...
//LIBSET   DD   DSNAME=LPR,DISP=OLD,...
//LIB      DD   DSNAME=ALIB,DISP=OLD,...
//         DD   DSNAME=BLIB,DISP=OLD,...
//SYSLIN   DD   *
 INCLUDE  SEQSET,LIBSET(SIM,ADD)
 ENTRY BEGIN
 NAME LMOD1
 INCLUDE LIBSET(DUALSIM),LIB(A1,A2,B3)
 ENTRY B3
 NAME LMOD2
/*
```

При создании модуля LMOD1 используется последовательный набор данных SET и два раздела библиотеки LPR, а при создании модуля LMOD2 — четыре раздела библиотек LPR, ALIB и BLIB.

Управляющее предложение

LIBRARY список наборов данных

указывает дополнительные библиотеки автовызова, разделы которых просматриваются для разрешения внешних ссылок до просмотра библиотеки, заданной редактору в наборе данных с dd-именем SYSLIB. Элементы списка наборов данных задают dd-имена библиотек и разделы в них в форме

dd-имя (список разделов)

Предложение LIBRARY может иметь также операнды в форме

(список внешних ссылок)

или

*(список внешних ссылок)

В списке перечислены внешние ссылки, которые не следует разрешать при помощи библиотек автовызова в пункте задания (первая форма) или в течение всего задания (вторая форма). Элементы списков наборов данных и внешних ссылок разделяют запятыми. В одном предложении LIBRARY можно использовать все формы операндов. Предложение LIBRARY нельзя применять, если указана опция NCAL.

Пример 4. Использование предложения LIBRARY.

```
//SYSLIB   DD   DSNAME=ALIB,DISP=OLD,...
//LIB      DD   DSNAME=BLIB,DISP=OLD,...
//SYSLIN   DD   DSNAME=OBJMOD,DISP=OLD,...
//         DD   *
 LIBRARY  LIB(BEG,WOT),(EX,WX)
/*
```

Для разрешения внешних ссылок путем автовызова библиотек сначала просматриваются разделы BEG и WOT библиотеки BLIB, а потом — библиотека ALIB. Внешние ссылки EX и WX при этом разрешаться не должны.

Управляющее предложение

CHANGE список замен

указывает внешние символы, которые нужно заменить, и их новые значения. Элементы списка замен разделяют запятыми. Они имеют форму

внешний символ(новый символ)

Предложение CHANGE должно находиться непосредственно перед модулем, в котором делаются замены, или непосредственно перед указывающим его предложением INCLUDE. В частности, это означает, что если последовательный набор данных содержит два модуля и нужно заменить внешнюю ссылку во втором по порядку модуле, то при помощи предложения CHANGE это сделать нельзя. Действие предложения CHANGE распространяется только на первый модуль последовательного набора данных.

Пример 5. Замена имени секции.

Объектные модули, подлежащие редактированию, находятся в последовательном наборе данных с dd-именем SYSLIN, созданным в пункте задания ST1. Первый объектный модуль содержит программную секцию с именем IHENTRY. При редактировании требуется заменить имя этой секции на ENTRY1.

Решение:

```
//OBJMOD   DD   DSNAME=*.ST1.SYSLIN,DISP=(OLD,DELETE)
//SYSLIN   DD   *
  CHANGE   IHENTRY(ENTRY1)
  INCLUDE  OBJMOD
/*
```

Управляющее предложение

REPLACE список замен

указывает либо программные секции, которые нужно заменить, и имена заменяющих их секций, либо имена исключаемых программных секций, либо имена исключаемых точек входа. Элементы списка замен имеют соответственно одну из трех форм:

```
имя программной секции(имя новой секции)
имя программной секции
имя точки входа
```

Если заменяется программная секция, то ссылки к имени старой секции внутри вводимого модуля заменяются ссылками к имени новой секции. При исключении секции или имени входа ссылки к соответствующему имени внутри модуля заменяются внешними ссылками, а если ссылок к исключаемому имени нет, то имя вычеркивается из словаря ESD.

Как и предложение CHANGE, предложение REPLACE должно находиться непосредственно перед модулем, в котором делаются замены, или перед предложением INCLUDE, указывающим набор данных, в котором находится этот модуль, причем изменяемый модуль должен быть первым в наборе данных.

Заметим, что в случае, когда объединяемые объектные или загрузочные модули содержат программные секции с одинаковыми именами, редактор связей включает в загрузочный модуль только первую из встретившихся ему одноименных секций. Все последующие секции с тем же именем игнорируются. Если нужно объе-

динить два модуля, в которых имеются программные секции с одинаковыми именами, то одну из одноименных секций нужно либо переименовать, либо исключить. Для исключения секции можно использовать предложение REPLACE.

Пример 6. Включение в загрузочный модуль программной секции.

Созданный в результате пакетной компиляции модулей Фортрана набор данных с dd-именем SYSLIN (пункт задания ST, пункт процедуры трансляции FORT) содержит два объектных модуля с программной секцией MAIN. Требуется: а) включить в загрузочный модуль программную секцию MAIN из первого объектного модуля; б) включить программную секцию MAIN из второго объектного модуля.

Решение:

```
а) //OBJMOD     DD   DSNAME=*.ST.FORT.SYSLIN,DISP=OLD.
   //SYSLIN     DD   *
    INCLUDE     OBJMOD
   /*
б) //OBJMOD     DD   DSNAME=*.ST.FORT.SYSLIN,DISP=OLD
   //SYSLIN     DD   *
     REPLACE    MAIN
     INCLUDE    OBJMOD
   /*
```

В случае (*а*) вторая программная секция с именем MAIN будет исключена редактором связей автоматически, а в случае (*б*) первая программная секция с именем MAIN исключается предложением REPLACE, поэтому вторая программная секция с тем же именем будет включена в загрузочный модуль.

4.2.7. Каталогизированные процедуры редактирования

В ОС ЕС имеется несколько каталогизированных процедур, включающих выполнение программы редактора связей. Здесь рассматриваются те из них, которые заканчиваются созданием загрузочного модуля.

Каталогизированная процедура редактирования LKED, текст которой приведен в п. 2.4.2 (пример 3), состоит из одного пункта (выполнение программы редактора связей). Каталогизированные процедуры трансляции и редактирования состоят из двух пунктов: в первом выполняется трансляция, а во втором — редактирование. Такие процедуры имеются для Ассемблера и компиляторов Алгола-60, Фортрана и ПЛ/1. Имена этих процедур заканчиваются буквами CL (Compilation, Link Edit — компиляция, редактирование связей): ASMFCL, ALGOFCL, FORTGCL и FORTHCL, PL1LFCL.

В пункте трансляции перечисленных процедур не описан входной набор данных трансляторов, содержащий исходный модуль. Этот набор данных добавляют к процедуре оператором DD вида

```
//имя пункта трансляции. SYSIN DD параметры набора данных
```

В пункте редактирования LKED процедур набор данных основного ввода редактора связей (dd-имя SYSLIN) состоит из двух сцепленных наборов данных. Первый набор данных содержит объ-

ектный модуль, созданный транслятором, а второй определяется параметром DDNAME вида

DDNAME=SYSIN

позволяющим добавлять к основному вводу управляющие предложения редактора связей и объектные модули оператором DD такого вида

//LKED.SYSIN DD параметры добавляемого набора данных

Для компиляторов каталогизированные процедуры трансляции и редактирования содержат описание библиотек подпрограмм (dd-имя SYSLIB), используемых в режиме автовызова. Для Ассемблера, напротив, автовызов запрещен опцией NCAL.

Каталогизированные процедуры трансляции и редактирования используют либо при отладке для проверки правильности межмодульных связей либо для создания загрузочных модулей с записью их в библиотеку. При вводе исходного модуля и управляющих предложений с перфокарт типичное задание выглядит так:

```
//имя задания JOB параметры
//имя пункта задания EXEC имя процедуры,параметры
//имя пункта трансляции.SYSIN  DD *
```

исходный модуль

```
/*
//LKED.SYSIN  DD *
```

управляющие предложения редактора

```
/*
```

Модифицирующие и добавляемые к пунктам процедуры операторы DD помещают перед соответствующими операторами DD *.

Ниже приведены тексты каталогизированных процедур трансляции и редактирования. Напомним, что при генерации системы или в процессе ее эксплуатации эти тексты могут быть изменены.

Каталогизированная процедура ASMFCL:

```
//ASM        EXEC  PGM=IEUASM,PARM=LOAD,REGION=5ØK
//SYSLIB     DD    DSNAME=SYS1.MACLIB,DISP=SHR
//SYSUT1     DD    DSNAME=&SYSUT1,UNIT=SYSSQ,                    *
//                 SPACE=(17ØØ,(4ØØ,5Ø)),SEP=(SYSLIB)
//SYSUT2     DD    DSNAME=&SYSUT2,UNIT=SYSSQ,                    *
//                 SPACE=(17ØØ,(4ØØ,5Ø))
//SYSUT3     DD    DSNAME=&SYSUT3,SPACE=(17ØØ,(4ØØ,5Ø)),         *
//                 UNIT=(SYSSQ,SEP=(SYSUT2,SYSUT1,SYSLIB))
//SYSPRINT   DD    SYSOUT=A
//SYSPUNCH   DD    SYSOUT=B
//SYSGO      DD    DSNAME=&LOADSET,UNIT=SYSSQ,                   *
//                 SPACE(8Ø,(2ØØ,5Ø)),DISP=(MOD,PASS)
//LKED       EXEC  PGM=IEWL,PARM=(XREF,LIST,NCAL),               *
//                 REGION=96K,COND=(8,LT,ASM)
//SYSLIN     DD    DSNAME=&LOADSET,DISP=(OLD,DELETE)
//          DD     DDNAME=SYSIN
//SYSLMOD    DD    DSNAME=&GOSET(GO),DISP=(MOD,PASS),            *
//                 UNIT=SYSDA,SPACE=(1Ø24,(5Ø,2Ø,1))
```

```
//SYSUT1      DD   DSNAME=&SYSUT1,SPACE=(1Ø24,(5Ø,2Ø)),                *
//               UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD))
//SYSPRINT    DD   SYSOUT=A
```

Каталогизированная процедура ALGOFCL:

```
//ALGOL       EXEC PGM=ALGOL,REGION=48K
//SYSPRINT    DD   SYSOUT=A
//SYSPUNCH    DD   SYSOUT=B
//SYSLIN      DD   DSN=&LOADSET,DISP=(MOD,PASS),                       *
//               UNIT=SYSSQ,SEP=SYSPUNCH,                             *
//               SPACE=(36ØØ,(1Ø,4))
//SYSUT1      DD   DSN=&SYSUT1,UNIT=SYSSQ,                             *
//               SEP=SYSPRINT,SPACE=(1Ø24,(5Ø,1Ø))
//SYSUT2      DD   DSN=&SYSUT2,UNIT=SYSSQ,                             *
//               SEP=SYSUT1,SPACE=(1Ø24,(5Ø,1Ø))
//SYSUT3      DD   DSN=&SYSUT3,UNIT=SYSDA,                             *
//               SPACE=(1024,(4Ø,1Ø))
//LKED        EXEC PGM=IEWL,PARM='XREF,LIST,LET',                      *
//               COND=(5,LT,ALGOL),REGION=96K
//SYSPRINT    DD   SYSOUT=A
//SYSLIN      DD   DSN=&LOADSET,DISP=(OLD,DELETE)
//            DD   DDNAME=SYSIN
//SYSLIB      DD   DSN=SYS1.ALGLIB,DISP=SHR
//SYSLMOD     DD   DSN=&GOSET(GO),DISP=(MOD,PASS),                     *
//               UNIT=SYSDA, SPACE=(1Ø24,(5Ø,2Ø,1))
//SYSUT1      DD   DSN=&SYSUT1,SEP=(SYSLIB,SYSLMOD),                   *
//               UNIT=SYSDA,SPACE=(1Ø24,(5Ø,2Ø))
```

Каталогизированная процедура FORTGCL:

```
// FORT       EXEC PGM=IEYFORT,REGION=1ØØK
//SYSPRINT    DD   SYSOUT=A
//SYSPUNCH    DD   SYSOUT=B
//SYSLIN      DD   DSNAME=&LOADSET,UNIT=SYSSQ,                         *
//               DISP=(MOD,PASS),DCB=BLKSIZE=8Ø,                      *
//               SPACE=(8Ø,(2ØØ, 1ØØ),RLSE)
//LKED        EXEC PGM=IEWL,PARM=(XREF,LET,LIST),                      *
//               COND=(4,LT,FORT),REGION=96K
//SYSLIB      DD   DSNAME=SYS1.FORTLIB,DISP=SHR
//SYSLMOD     DD   DSNAME=&GOSET(MAIN),UNIT=SYSDA,                     *
//               DISP=(NEW,PASS),DCB=BLKSIZE=1Ø24,                    *
//               SPACE=(1Ø24,(2Ø,1Ø,1),RLSE)
//SYSPRINT    DD   SYSOUT=A
//SYSUT1      DD   DSNAME=&SYSUT1,DCB=BLKSIZE=1Ø24,                    *
//               UNIT=SYSDA,SPACE=(1Ø24,(1ØØ,1Ø),RLSE)
//SYSLIN      DD   DSNAME=&LOADSET,DISP=(OLD,DELETE)
//            DD   DDNAME=SYSIN
```

Каталогизированная процедура FORTHCL:

```
//FORT        EXEC PGM=IEKAØØ,REGION=228K
//SYSPRINT    DD   SYSOUT=A
//SYSPUNCH    DD   SYSOUT=B
//SYSLIN      DD   DSNAME=&LOADSET,UNIT=SYSSQ,                         *
//               DISP=(MOD,PASS),SPACE=(4ØØ,(2ØØ,5Ø),RLSE)
//LKED        EXEC PGM=IEWL,PARM=(MAP,LET,LIST),                       *
//               COND=(4,LT,FORT),REGION=96K
//SYSLIB      DD   DSNAME=SYS1.FORTLIB,DISP=SHR
//SYSPRINT    DD   SYSOUT=A
//SYSLMOD     DD   DSNAME=&GOSET(MAIN),UNIT=SYSDA,                     *
```

```
//                      DISP=(,PASS),SPACE=(3072,(30,10,1),RLSE)
//SYSLIN        DD      DSNAME=&LOADSET,DISP=(OLD,DELETE)
//              DD      DDNAME=SYSIN
//SYSUT1        DD      DSNAME=&SYSUT1,UNIT=SYSDA,                    *
//                      SPACE=(1024,(200,20)),SEP=SYSLMOD
```

Каталогизированная процедура PL1LFCL:

```
//PL1L          EXEC  PGM=IEMAA,PARM='LOAD,NODECK',                   *
//                    REGION=52K
//SYSPRINT      DD    SYSOUT=A
//SYSLIN        DD    DSNAME=&&LOADSET,DISP=(MOD,PASS),               *
//                    UNIT=SYSSQ,SPACE=(80,(250,100))
//SYSUT3        DD    DSNAME=&&SYSUT3,DCB=BLKSIZE=80,                 *
//                    UNIT=SYSDA,SPACE=(80,(250,250))
//SYSUT1        DD    DSNAME=&&SYSUT1,DCB=BLKSIZE=1024,               *
//                    UNIT=SYSDA,SEP=(SYSUT3,SYSLIN),                 *
//                    SPACE=(1024,(60,60),,CONTIG)
//LKED          EXEC  PGM=IEWL,PARM='XREF,LIST',                      *
//                    COND=(9,LT,PL1L),REGION=96K
//SYSLIB        DD    DSNAME=SYS1.PL1LIB,DISP=SHR
//SYSLMOD       DD    DSNAME=&&GOSET(GO),UNIT=SYSDA,                  *
//                    DISP=(MOD,PASS),SPACE=(1024,(50,20,1),RLSE)
//SYSUT1        DD    DSNAME=&&SYSUT1,UNIT=SYSDA,                     *
//                    SPACE=(1024,(200,20)),                          *
//                    SEP=(SYSLMOD,SYSLIB),DCB=BLKSIZE=1024
//SYSPRINT      DD    SYSOUT=A
//SYSLIN        DD    DSNAME=&&LOADSET,DISP=(OLD,DELETE)
//              DD    DDNAME=SYSIN
```

Пример 1. Пакетная трансляция и редактирование модулей Фортрана.

```
//EXF JOB (1976—38),'ИВАНОВ И.И.',MSGLEVEL=1
//STF EXEC FORTGCL
//FORT.SYSIN DD *
```

исходный модуль Фортрана 1

исходный модуль Фортрана 2

```
/*
```

Если задание на этом заканчивается, то после его завершения созданный загрузочный модуль будет уничтожен вместе с содержащей его временной библиотекой GOSET.

Пример 2. Трансляция и создание загрузочного модуля ПЛ/1 с записью его в существующую сохраняемую библиотеку.

```
//EXPL JOB (1976—381),'ПЕТРОВ П.П.',MSGLEVEL=1
//STPL EXEC PL1LFCL,PARM.PL1L = 'N=MOD6,OPT=2'
//PL1L.SYSIN DD *
```

исходный модуль ПЛ/1

```
/*
//LKED.SYSLMOD DD DSN=LIBCAF,DISP=(OLD,PASS),                         *
//       UNIT=5050,VOL=SER=LIBPAK
```

Опции компилятора ПЛ/1 LOAD и NODECK в пункте PL1L процедуры будут подавлены. Однако эти опции останутся действующими по умолчанию (см. п. 3.1.3), если в системе при ее генерации не установлены другие стандартные значения опций.

Опция N=MOD6 дает имя загрузочному модулю. Напомним, что N — сокращение OBJNM (см. п. 3.1.3). Применение этой опции в данном случае эквивалентно добавлению к заданию операторов.

```
//LKED.SYSIN DD *
  NAME MOD6(R)
/*
```

Следовательно, загрузочный модуль будет записан в библиотеку LIBCAF под именем MOD6, а если в библиотеке уже есть модуль с таким именем, то он будет заменен вновь созданным модулем.

Пример 3. Пакетная компиляция модулей ПЛ/1 с редактированием и занесением загрузочного модуля в сохраняемую библиотеку LIBCAF.

```
//EXPL1 JOB (1976—382),'ВЕТРОВ',MSGLEVEL=1
//STPL1 EXEC PL1LFCL,PARM.PL1L='LOAD,NODECK'
//PL1L.SYSIN DD *
```

исходный модуль ПЛ/1 A

```
* PROCESS ('LOAD,NODECK');
```

исходный модуль ПЛ/1 B

```
/*
//LKED.SYSLMOD DD DSN=LIBCAF,DISP=(OLD,KEEP),                    *
//         UNIT=5050,VOL=SER=LIBPAK
//LKED.SYSIN DD *
       NAME MOD7(R)
/*
```

Если не требуется записывать загрузочный модуль в сохраняемую библиотеку, то операторы с dd-именами LKED.SYSLMOD и LKED.SYSIN, а также предложение NAME нужно опустить. В этом случае загрузочный модуль будет помещен во временную библиотеку &&GOSET под именем GO (см. текст каталогизированной процедуры PL1LFCL).

4.3. ВЫПОЛНЕНИЕ ЗАГРУЗОЧНОГО МОДУЛЯ

4.3.1. Вызов

Загрузочный модуль можно вызвать для выполнения, указав его имя (или алиас) в параметре PGM оператора EXEC или в макрокоманде динамического вызова. Если вызываемый модуль создан в другом задании, то он должен находиться либо в общей системной библиотеке SYS1.LINKLIB, либо в личной библиотеке задания или пункта задания. (Пример использования в задании личных библиотек приведен в п. 2.3.6.)

Загрузочный модуль, созданный в том же задании, обычно указывают не именем, а ссылкой на библиотеку, в которой он находится. В этом случае параметр PGM записывают в одной из двух форм:

```
PGM=*.имя пункта задания.dd-имя библиотеки
PGM=*.имя пункта задания.имя пункта процедуры.dd-имя библиотеки
```

Вторую форму используют, если библиотека создана или описана в процедуре. Например, загрузочный модуль, созданный при выполнении каталогизированной процедуры FORTGCL в примере 1 п. 4.2.7, можно вызвать в том же задании оператором

```
//имя пункта EXEC PGM=*.STF.LKED.SYSLMOD
```

После оператора EXEC нужно записать операторы DD, описывающие наборы данных, которые использует загрузочный модуль.

4.3.2. Наборы данных

В программе на языке Ассемблера dd-имена наборов данных, с которыми работает программа, указывают в макрокомандах DCB.

Языки высокого уровня не используют понятие «набор данных». Оно относится к операционной системе ОС ЕС. При составлении задания абстрактные совокупности данных, описанные в программе языковыми средствами (файлы, вводимые и выводимые данные), нужно отобразить на наборы данных операционной системы. Правила этого отображения устанавливаются соглашениями, связывающими языковые понятия конкретного языка с dd-именами наборов данных ОС ЕС. Для разных языков программирования эти соглашения различны.

В процедурах ввода-вывода Алгола-60 используют условные номера наборов данных. Между этими номерами и dd-именами наборов данных загрузочного модуля установлено постоянное соответствие: номер 0 — dd-имя SYSIN, любой другой номер 1, 2, ..., 15 — dd-имя ALGLDDØ1, ALGLDDØ2, ..., ALGLDD15. По стандартному соглашению набор данных с dd-именем SYSIN применяют для ввода данных, набор данных с dd-именем ALGLDDØ1 — для вывода данных на АЦПУ. Остальные наборы данных можно использовать для любых целей.

Кроме наборов данных общего назначения, загрузочный модуль Алгола-60 использует специальные наборы данных со следующими dd-именами: SYSPRINT — для листинга загрузочного модуля, SYSUT1 — для записи трассы программы при отладке (см. разд. 5.1), SYSUT2 — для промежуточной записи и выборки данных во время выполнения программы без преобразования данных во внешний формат. Наборы данных SYSUT1 и SYSUT2 создают на магнитной ленте или на диске.

В каталогизированных процедурах Алгола-60 для загрузочного модуля описаны только наборы данных ALGLDDØ1, SYSPRINT и SYSUT1.

Операторы ввода-вывода Фортрана, как и процедуры ввода-вывода Алгола-6Ø, содержат условные номера наборов данных, называемые *ссылочными номерами наборов данных*. Эти номера

включают в dd-имена наборов данных загрузочного модуля следующим образом. Любое dd-имя загрузочного модуля Фортрана пишут в форме

FTxxFyyy

Здесь xx — ссылочный номер набора данных в операторе ввода-вывода, yyy — порядковый номер набора данных, используемый для индивидуализации dd-имен, имеющих один и тот же ссылочный номер. Обычно полагают yyy=001. Значения порядкового номера 002,003, ... можно использовать в dd-имени только в специальных случаях, например при записи последовательных наборов данных на непомеченную магнитную ленту или при чтении нескольких разделов одной библиотеки. Порядковые номера наборов данных бо́льшие единицы образуются только динамически при взаимодействии утверждения ENDFILE с утверждением WRITE (при записи нескольких наборов данных на непомеченную ленту), а также при выполнении утверждения READ, использующего выход END и взаимодействующего с утверждениями READ или WRITE, в которых употребляется тот же ссылочный номер. Подчеркнем еще раз, что, кроме отмеченных выше специальных случаев, всегда нужно полагать значение порядкового номера равным 001.

В каталогизированных процедурах Фортрана для загрузочного модуля используют следующие стандартные dd-имена: FT05F001 — для ввода данных, FT06F001 — для вывода данных на АЦПУ, FT07F001 — для вывода данных на перфокарты. Естественно, к процедуре всегда можно добавить дополнительные операторы DD или модифицировать имеющиеся. Однако если программист желает сократить задание, то он может воспользоваться в своей программе указанными стандартными ссылочными номерами: 5 — для ввода данных, 6 — для печати и 7 — для вывода на перфокарты.

В исходной программе на Коболе в разделе ENVIRONMENT DIVISION утверждение SELECT содержит имя файла, а утверждение ASSIGN — внешнее имя. Эти же имена нужно указать в задании в качестве dd-имен соответствующих им наборов данных.

В программах на языке ПЛ/1 при неявном открытии файлов утверждениями GET и PUT по умолчанию открываются стандартные файлы SYSIN (для ввода данных) и SYSPRINT (для вывода). Этим стандартным файлам соответствуют стандартные dd-имена SYSIN и SYSPRINT. Набор данных с dd-именем SYSPRINT описан в каталогизированных процедурах ПЛ/1 и определяет вывод на печать.

При явном открытии файлов имена файлов исходной программы на ПЛ/1 связываются с dd-именами соответствующих им наборов данных при помощи опции TITLE утверждения OPEN. Если утверждение OPEN не содержит этой опции, то компилятор образует dd-имя из первых восьми литер имени файла. Такое же dd-имя нужно записать в задании.

. Для программы на языке РПГ наборы данных создаются для всех файлов, определенных в бланке описания файлов. Их dd-имена должны совпадать с именами файлов на бланке. В каталогизированных процедурах РПГ наборы данных для загрузочного модуля не определены. Программист всегда добавляет описания этих наборов данных сам.

4.3.3. Каталогизированные процедуры выполнения

Для всех трансляторов в ОС ЕС имеются каталогизированные процедуры трансляции, редактирования и выполнения. Каждая из них состоит из трех пунктов: трансляции, редактирования и выполнения. Имена этих процедур заканчиваются буквами CLG (Compilation, Link Edit, Go): ASMFCLG, ALGOFCLG, COBUCLG, FORTGCLG, FORTHCLG, PL1LFCLG, RPGECLG.

При использовании этих процедур нужно указать: для пункта трансляции — исходный модуль, для пункта редактирования LKED — управляющие предложения редактора связей (иногда также объектные модули), а для пункта выполнения GO — исходные данные и дополнительные наборы данных загрузочного модуля. При вводе исходного модуля, управляющих предложений и исходных данных с перфокарт типичное задание может выглядеть так:

```
//имя задания JOB параметры
//имя пункта задания EXEC имя процедуры,параметры
//имя пункта трансляции.SYSIN DD *
                    | исходный модуль |
/*
//LKED.SYSIN DD *
          | управляющие предложения редактора |
/*
//GO.SYSIN DD *
     | исходные данные для загрузочного модуля |
/*
```

Если нужны дополнительные наборы данных загрузочного модуля, то их добавляют к пункту GO. Все модифицирующие и добавляемые к процедуре операторы DD помещают перед соответствующими операторами DD *. Ниже приведены тексты каталогизированных процедур.

Каталогизированная процедура ASMFCLG:

```
//ASM       (см. пункт ASM процедуры ASMFCL в п. 4.2.7)
//LKED      (см. пункт LKED процедуры ASMFCL в п. 4.2.7)
//GO EXEC   PGM=*.LKED.SYSLMOD,                          *
//          COND=((8,LT,ASM),(4,LT,LKED))
```

Каталогизированная процедура ALGOFCLG:

```
//ALGOL          (см. пункт ALGOL процедуры ALGOFCL в п. 4.2.7)

//LKED           (см. пункт LKED процедуры ALGOFCL в п. 4.2.7)

//GO  EXEC  PGM=*.LKED.SYSLMOD,                                    *
//          COND=((5,LT,ALGOL),(5,LT,LKED))
//ALGLDDØ1  DD  SYSOUT=A
//SYSPRINT  DD  SYSOUT=A
//SYSUT1    DD  DSN=&SYSUT1,UNIT=SYSSQ,                            *
//              SPACE= (1Ø24, (2Ø,1Ø))
```

Каталогизированная процедура COBUCLG:

```
//COB  EXEC  PGM=IKFCBLØØ,REGION=86K,                              *
//           PARM='LOAD,SUPMAP'
//SYSPRINT DD SYSOUT=A
//SYSUT1  DD  UNIT=SYSDA,SPACE=(46Ø,(7ØØ,1ØØ))
//SYSUT2  DD  UNIT=SYSDA,SPACE=(46Ø, (7ØØ,1ØØ))
//SYSUT3  DD  UNIT=SYSDA,SPACE=(46Ø, (7ØØ,1ØØ))
//SYSUT4  DD  UNIT=SYSDA,SPACE=(46Ø(7ØØ,1ØØ))
//SYSLIN  DD  DSNAME=&LOADSET,DISP=(MOD,PASS),                     *
//            UNIT=SYSDA,SPACE=(8Ø,(5ØØ,1ØØ))
//LKED EXEC PGM=IEWL,PARM='LIST,XREF,LET',                         *
//          COND=(5,LT,COB),REGION=96K
//SYSLIN  DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)
//        DD  DDNAME=SYSIN
//SYSLMOD DD DSNAME=&GODATA(RUN),UNIT= SYSDA,                      *
//           DISP=(NEW,PASS),SPACE=(1Ø24,(5Ø,2Ø,1))
//SYSLIB  DD  DSNAME=SYS1.COBLIB,DISP=SHR
//SYSUT1  DD  UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),                   *
//            SPACE=(1Ø24,(5Ø,2Ø))
//SYSPRINT DD SYSOUT=A
//GO  EXEC PGM=*.LKED.SYSLMOD,                                     *
//         COND=((5,LT,COB),(5,LT,LKED))
```

Каталогизированная процедура FORTGCLG:

```
//FORT  (см. пункт FORT процедуры FORTGCL в п. 4.2.7)

//LKED  (см. пункт LKED процедуры FORTGCL в п. 4.2.7)

//GO    EXEC PGM=*.LKED.SYSLMOD,                                   *
//           COND=((4,LT,FORT),(4,LT,LKED))
//FT05FØØ1 DD DDNAME=SYSIN
//FT06FØØ1 DD SYSOUT=A
//FT07FØØ1 DD SYSOUT=B
```

Каталогизированная процедура FORTHCLG:

```
//FORT  (см. пункт FORT процедуры FORTHCL в п. 4.2.7)

//LKED  (см. пункт LKED процедуры FORTHCL в п. 4.2.7)

//GO    EXEC PGM=*.LKED.SYSLMOD,                                   *
//           COND=((4,LT,FORT),(4,LT,LKED))
//FT05FØØ1 DD DDNAME=SYSIN
//FT06FØØ1 DD SYSOUT=A
//FT07FØØ1 DD SYSOUT=B
```

Каталогизированная процедура PL1LFCLG:

```
//PL1L  (см. пункт PL1L процедуры PL1LFCL в п. 4.2.7)
.
.
.
//LKED  (см. пункт LKED процедуры PL1LFCL в п. 4.2.7)
.
.
.
//GO     EXEC PGM=*.LKED.SYSLMOD,                                    *
//            COND=((9,LT,LKED),(9,LT,PL1L))
//SYSPRINT DD SYSOUT=A
```

Каталогизированная процедура RPGECLG:

```
//DEFAULT  PROC RGREGN=52K,                                          *
//         RGPARM='NODECK,LOAD,LIST,NOSEQN',                         *
//         RGU3SPC='(600,(100,20))',                                 *
//         RGU2SPC='(600,(100,20))',                                 *
//         RGU1SPC='(600,(100,20))',                                 *
//         RGGOSPC='(80,(200,50))',                                  *
//         LKREGN=100K,LKPARM='XREF,LIST,LET',                       *
//         LKMDSPC='(1024,(50,20,1))',                               *
//         LKU1SPC='(1024,(50,20))',                                 *
//         GOCOND='((9,LT,RPG),(5,LT,LKED))',                        *
//         GOPARM='DATE=000000'
//RPG      EXEC PGM=IESRPG,REGION=&RGREGN,                           *
//              PARM=(&RGPARM)
//SYSPRINT  DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSUT3 DD DSNAME=&&SYSUT3,UNIT=SYSSQ,                              *
//          SPACE=&RGU3SPC
//SYSUT2 DD  DSNAME=&&SYSUT2,UNIT=SYSSQ,                             *
//          SPACE=&RGU2SPC
//SYSUT1 DD DSNAME=&&SYSUT1,UNIT=SYSSQ,                              *
//          SPACE=&RGU1SPC
//SYSGO  DD DSNAME=&&LOADSET,UNIT=(SYSSQ,                            *
//           SEP=SYSPUNCH),DISP=(MOD,PASS,                           *
//          DELETE),SPACE=&RGGOSPC
//LKED EXEC PGM=IEWL,REGION=&LKREGN,                                 *
//          PARM=(&LKPARM)                                           *
//SYSLIN DD DSNAME=&&LOADSET,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//SYSLMOD DD DSNAME=&&GOSET(RPG),UNIT=SYSDA,                         *
//           DISP=(NEW,PASS,DELETE),SPACE=&LKMDSPC
//SYSUT1 DD DSNAME=&&SYSUT1,SPACE=&LKU1SPC,                          *
//          UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD))
//SYSPRINT DD SYSOUT=A
//GO     EXEC PGM=*.LKED.SYSLMOD,                                    *
//           COND=&GOCOND,PARM='&GOPARM'
```

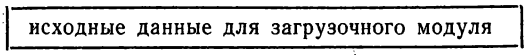
Пример 1. Пакетная трансляция, редактирование и выполнение модулей Фортрана.

```
//EXF1  JOB   (1976—383),'C.A.',MSGLEVEL=1
//STF1  EXEC  FORTGCLG
//FORT.SYSIN DD  *
```

исходный модуль Фортрана 1

исходный модуль Фортрана 2

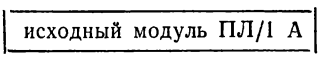
```
/*
//GO.SYSIN  DD  *
```



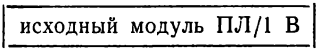

```
/*
```

Пример 2. Пакетная трансляция, редактирование и выполнение модулей ПЛ/1.

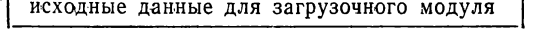
```
//PLCLG JOB    (1976—384),'C.A.',MSGLEVEL=1
//STCLG EXEC PL1LFCLG
//PL1L.SYSIN   DD  *
```




```
* PROCESS;
```




```
/*
//GO.SYSIN   DD  *
```

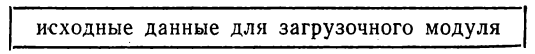



```
/*
```

Для компиляторов Кобола, Фортрана, ПЛ/1 и транслятора РПГ имеются также каталогизированные процедуры редактирования и выполнения: COBULG, FORTGLG, FORTHLG, PL1LFLG и RPGELG. В них нет пункта трансляции. Эти процедуры обычно применяют для редактирования и выполнения таких загрузочных модулей, которые постоянно хранятся в библиотеках и не содержат библиотечных подпрограмм.

Пример 3. Редактирование и выполнение загрузочного модуля ПЛ/1, хранящегося в сохраняемой библиотеке LIBCAF.

```
//EXPLLG JOB    (1976—42),'C.A.',MSGLEVEL=2
//STPLLG EXEC PL1LFLG
//LKED.SYSIN DD DSN=LIBCAF(MOD6),                                    *
//              DISP=(OLD,KEEP),UNIT=5Ø5Ø,VOL=SER=LIBPAK
//GO.SYSIN   DD  *
```




```
/*
```

4.4. ЗАГРУЗЧИК

4.4.1. Функции загрузчика

Большой объем программы редактора связей ОС ЕС существенно замедляет процесс редактирования. Для небольших программ время редактирования сравнимо со временем трансляции, но при обработке программ простой структуры все возможности редактора не используются. С другой стороны, нередко сразу после редактирования нужно выполнить полученный загрузочный модуль. В этом случае промежуточная запись загрузочного модуля в библиотеку на диске, последующая загрузка модуля в основную память и повторная корректировка адресных констант программой выборки оказываются излишними операциями. К тому

же, вынужденное разделение редактирования и выполнения на два пункта задания приводит к дополнительным затратам времени в системе на работу управляющей программы.

Для уменьшения затрат машинного времени в случаях, когда редактирование непосредственно предшествует выполнению, вместо редактора можно использовать загрузчик. *Загрузчик* — это системная обрабатывающая программа, объединяющая выполнение основных функций редактора связей и программы выборки в одном пункте задания. Загрузчик загружает находящиеся в его входном наборе данных объектные и загрузочные модули в основную память, объединяет их в единую программу, корректируя перемещаемые адресные константы с учетом фактического адреса загрузки, и передает управление в точку входа полученной программы.

Если после редактирования в созданной программе остались неразрешенные внешние ссылки, то для их разрешения в мультипрограммных режимах загрузчик может организовать поиск недостающих модулей в общей области резидента, где находятся часто используемые разными программами подпрограммы (в однопрограммном режиме общая область резидента не используется). Затем загрузчик может продолжить поиск в указанных ему библиотеках автовызова.

Загрузчик выполняет не все функции редактора связей: он не обрабатывает управляющие предложения редактора и некоторые опции; не может создавать оверлейные программы. Обработанная загрузчиком программа выполняется немедленно. Ее нельзя записать в библиотеку.

4.4.2. Вызов загрузчика и его опции

Загрузчик постоянно хранится в библиотеке SYS1.LINKLIB, которая содержит все системные обрабатывающие программы. В библиотеке загрузчик занимает почти вдвое меньше места, чем редактор связей версии F44 и F88. Программу загрузчика можно вызвать, указав ее имя IEWDRGO (или алиас LOADER) в параметре PGM оператора EXEC или в макрокоманде динамического вызова. Для выполнения программы загрузчика нужно выделить основную память: 11К — для размещения программы загрузчика, 4К — для программ управления данными, не менее 2—8К — для буферов и таблиц (в частности, не менее 3К для Фортрана и не менее 8К для ПЛ/1), а также необходимую память для размещения обрабатываемой загрузчиком программы. Промежуточная внешняя память загрузчику не нужна.

Единственное средство управления работой загрузчика — его опции. Опции загрузчика и параметры, которые нужно передать сформированной им программе, записывают в параметре PARM оператора EXEC, вызывающего загрузчик, следующим образом:

PARM='список опций загрузчика/список параметров'

Элементы обоих списков разделяют запятыми. Знак «/» всегда предшествует списку параметров объектной программы, даже тогда, когда опции загрузчика отсутствуют. Этот знак не пишут лишь в случае, если список параметров пуст.

Ниже описаны опции загрузчика. Утвердительная и отрицательная формы разделены знаком «/». Пояснения относятся к утвердительной форме. Стандартные значения опций подчеркнуты.

MAP/NOMAP — напечатать список внешних имен и их адреса.

CALL/NOCALL — использовать библиотеки автовызова.

RES/NORES — для разрешения внешних ссылок до просмотра библиотек автовызова просмотреть общую область резидента (RES автоматически устанавливает CALL).

LET/NOLET — выполнить объектную программу даже при обнаружении в ходе ее формирования ошибок серьезности 2 (код возврата 8).

PRINT/NOPRINT — печатать диагностические сообщения.

SIZE=размер — указание размера области памяти, которую загрузчик может использовать для буферов, таблиц и размещения обрабатываемой программы (по умолчанию принимается SIZE=100K).

EP=имя — указание точки входа объектной программы (точки входа для Фортрана — MAIN, для ПЛ/1 — IHENTRY, для Алгола-60 — IHIFSAIN — стандартны, их можно явно не указывать, если модули, полученные трансляцией с этих языков, не объединяются с модулями, транслированными с других языков).

Пример. Использование опций загрузчика и указание параметров объектной программы

```
//LOAD  EXEC  PGM=LOADER,PARM='MAP/X,Y'
```

После оператора EXEC, вызывающего загрузчик, нужно написать операторы DD, определяющие наборы данных загрузчика и объектной программы.

4.4.3. Наборы данных загрузчика и объектной программы

Входной набор данных загрузчика (dd-имя SYSLIN) является обязательным. Он может содержать объектные или загрузочные модули (рис. 4.14). Объектные модули могут находиться в последовательных и библиотечных наборах данных, а загрузочные модули — только в библиотечных. Для библиотечных наборов данных нужно указать имя библиотеки и имя раздела. Чтобы включить во входной набор данных несколько модулей, можно использовать сцепленные операторы DD.

Библиотеки автовызова (dd-имя SYSLIB) нужны, если используются опции CALL или RES. Этот набор данных может состоять из нескольких сцепленных библиотек, но все библиотеки должны содержать либо только объектные, либо только загрузочные модули.

Диагностический выходной набор данных загрузчика (dd-имя SYSLOUT) нужен, если используется опция PRINT (или MAP). Этот набор данных должен быть последовательным. Он предназначен для листинга загрузчика.

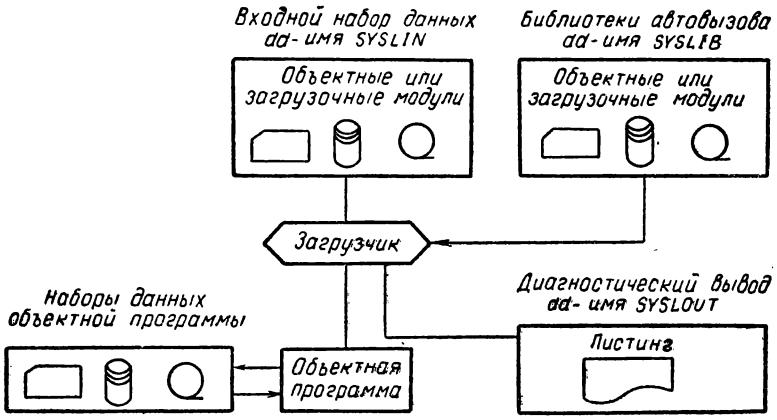


Рис. 4.14. Наборы данных загрузчика и объектной программы

В пункте задания, в котором выполняется программа загрузчика, должны быть определены также все наборы данных объектной программы. Операторы DD для них записывают в соответствии с соглашениями, описанными в п. 4.3.2 для наборов данных загрузочного модуля.

Пример. Трансляция, загрузка и выполнение модулей Фортрана.

```
//EXLOF   JOB    (1976—387),'БОРИСОВ Б. Б.'
//STLOF   EXEC   FORTCG
//FORT.SYSIN DD      *
```

исходный модуль Фортрана 1

исходный модуль Фортрана 2

```
/*
//LOADGO     EXEC  PGM=LOADER,PARM=MAP
//SYSLIB     DD    DSNAME=SYS1.FORTLIB,DISP=SHR
//SYSLIN     DD    DSNAME=*.STLOF.FORT.SYSLIN,DISP=OLD
//SYSLOUT    DD    SYSOUT=A
//FT06F001   DD    SYSOUT=A
//FT05F001   DD       *
```

исходные данные объектной программы

```
/*
```

4.4.4. Каталогизированные процедуры трансляции, загрузки и выполнения

Каталогизированные процедуры трансляции, загрузки и выполнения имеются только для Алгола-60, Кобола и ПЛ/1. Они имеют имена ALGOFCG, COBUCG и PL1LFCG. Эти процедуры состоят из двух пунктов. Пункты «трансляция» полностью копи-

руют тексты процедур трансляции ALGOFC, COBUC, PL1LFC соответственно. Пункты загрузки и выполнения (GO) включают стандартные наборы данных для загрузочного модуля (см. процедуры ALGOFCLG и PL1LFCLG в п. 4.3.3). В процедуре COBUCG таких наборов данных нет.

Для использования каталогизированных процедур трансляции, загрузки и выполнения к пункту «трансляция» нужно добавить входной набор данных транслятора, содержащий исходный модуль. Этот набор данных добавляют при помощи оператора DD вида

```
//имя пункта трансляции.SYSIN DD параметры входного набора
```

а к пункту «загрузка и выполнение» (пункт GO) — оператор DD, определяющий исходные данные объектной программы. Этот оператор DD имеет вид:

```
//GO.SYSIN DD параметры набора данных с исходными данными
```

Пример 1. Трансляция, загрузка и выполнение модуля Алгола-60.

```
//EXAL   JOB    (1976—388),'СЕМЕНОВ С.С.',MSGLEVEL=1
//STAL   EXEC   ALGOFCG
//ALGOL.SYSIN DD   *
```

исходный модуль Алгола-60

```
/*
//GO.SYSIN   DD   *
```

исходные данные объектной программы

```
/*
```

Пример 2. Пакетная трансляция, загрузка и выполнение модулей ПЛ/1.

```
//EXPLCG   JOB     (1976—389),'С.А.Ф.',MSGLEVEL=1
//STPLCG   EXEC    PL1LFCG
//PL1L.SYSIN  DD  *
```

исходный модуль ПЛ/1 А

```
*PROCESS;
```

исходный модуль ПЛ/1 В

```
/*
//GO.SYSIN   DD   *
```

исходные данные для объектной программы

```
/*
```

Оба исходные модуля — внешние процедуры ПЛ/1, причем одна из внешних процедур — главная (OPTIONS (MAIN)).

Глава 5

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О СИСТЕМЕ ПРОГРАММИРОВАНИЯ

5.1. СРЕДСТВА ОТЛАДКИ

5.1.1. Общесистемные средства

Программист расходует на отладку программы в среднем от 20 до 50% своего рабочего времени, что говорит о большом значении средств отладки в комплексе средств подготовки программ. В системе программирования ЕС ЭВМ можно выделить три группы средств отладки: общесистемные, средства языков программирования и средства трансляторов.

Общесистемные средства входят в состав системных программ и обеспечивают пользователей информационными, диагностическими и другими сообщениями, а также дампами.

Таблица 5.1

Префиксы системных программ

Префикс	Программа	Префикс	Программа
IBC	Независимые утилиты	IED	Неавтономная тестовая программа
IEA	Супервизор		
IEB	Утилиты для обработки наборов данных	IGC	Транзитные SVC-программы
IEC	Супервизор ввода-вывода	IGE	Программы обработки ошибок ввода-вывода
IEE	Главный планировщик		
IEF	Планировщик заданий	IGG	Программы обеспечения ввода-вывода
IEG	Тестран		
IEH	Системные утилиты	IHA	Системные управляющие блоки
IEI	Программы библиотеки генерации системы	IHB	Программы библиотеки макроопределений
IEK	Транслятор Фортран оптимизирующий	IHC	Программы библиотеки Фортрана
IEM	Транслятор ПЛ/1		
IKF	Транслятор Кобол	IHD	Программы библиотеки Кобола
IER	Программа СОРТИРОВКА-ОБЪЕДИНЕНИЕ	IHE	Программы библиотеки ПЛ/1
IES	Транслятор РПГ		
IEU	Ассемблер	IHF	Программы библиотеки ПЛ/1
IEW	Редактор, оверлейный супервизор, программа FETCH	IHG	Программа анализа и обновления
IEX	Транслятор Алгол-60	IHI	Программы библиотеки Алгола
IEY	Транслятор Фортран стандартный	IHJ	Программы КОНТРОЛЬНАЯ ТОЧКА-РЕСТАРТ

Сообщение, выдаваемое в листинге, состоит из идентификатора и текста сообщения. Префикс идентификатора (табл. 5.1) указывает системную программу, выдавшую сообщение, а номер идентифицирует данное сообщение. Номер может содержать указатель характера сообщения. При помощи идентификатора сообщения программист (оператор) находит необходимые ему пояснения и рекомендации в руководстве по соответствующей системной программе.

Пример 1. Сообщение планировщика заданий.

IEF208I INVALID REGION PARAMETER

Причина: в операторе JOB или EXEC параметр REGION определяет величину, превышающую величину динамической области памяти. Выполнение задания прекращается.

Пример 2. Сообщение

IEW0143 ERROR — NO TEXT

выдано редактором связей. Причина: в выходном модуле отсутствует текст; либо все программные секции вычеркнуты из ввода, либо нет программных секций, которые содержали бы текст. Редактор прекращает обработку.

Дамп представляет собой информацию о ходе выполнения программы и состоянии системы. По способу получения и характеру информации различают следующие виды дампов: дамп аварийного завершения (дамп ABEND), дамп снимков памяти (дамп SNAP), индикативный дамп, независимый дамп.

Дамп аварийного завершения состоит из управляющей информации и содержимого основной памяти, используемой при выполнении задания. Содержимое памяти печатается построчно. В каждой строке указывается шестнадцатеричный адрес первого байта памяти в строке, восемь слов (32 байта) содержимого памяти в шестнадцатеричном коде и те же восемь слов в ДКОИ (32 символа). Отсутствие в последнем символа, соответствующего какому-либо байту, отмечается в дампе точкой.

Описание управляющей информации можно найти в руководстве по отладке. Часто из всей управляющей информации достаточно знать только код завершения (COMPLETION CODE): системный или пользовательский. Код завершения указывает возможную причину аварийного завершения задания. Некоторые коды сопровождаются пояснениями.

Пример 3. Фрагмент управляющей информации дампа ABEND.

```
COMPLETION CODE — SYSTEM=0C1     USER=0000
PROGRAM INTERRAPTION (OPERATION) AT LOCATION 000028
```

Пояснения и рекомендации программисту и оператору для каждого системного кода завершения можно найти в перечне сообщений и кодов ОС ЕС.

Описываемый дамп обеспечивается автоматически выдаваемой системной макрокомандой ABEND. Программист должен только записать в задании оператор DD, определяющий набор данных для дампа, с dd-именем SYSABEND или SYSUDUMP (см. п. 2.3.2).

Напомним, что в дампе не будет представлено содержимое области ядра супервизора, если в качестве dd-имени указано SYSUDUMP.

Дамп снимков памяти выдается с помощью системной макрокоманды SNAP. Отличия его от дампа ABEND состоят в следующем:

программист сам записывает макрокоманду SNAP в программе и определяет для дампа выходной набор данных в задании (dd-имя не должно совпадать с SYSABEND и SYSUDUMP);

дамп содержит только ту информацию, которую программист указывает операндами макрокоманды;

после выдачи дампа выполнение задания продолжается.

Индикативный дамп автоматически выдается при ненормальном завершении задачи, если в задании не записан (или записан неверно) оператор DD с dd-именем SYSABEND или SYSUDUMP. В отличие от дампа ABEND индикативный дамп не содержит распечатки основной памяти.

Независимый дамп получают при помощи специальной программы, которая хранится на перфокартах и вводится процедурой начальной загрузки. Дамп отражает содержимое всей основной памяти. Им пользуются крайне редко.

5.1.2. Средства отладки в языках и трансляторах

Некоторые входные языки системы программирования ОС ЕС имеют собственные отладочные средства. Таких средств нет в РПГ, Алголе-60 и Фортране оптимизирующем. В Алголе-60 средства отладки включены в компилятор, а Фортран оптимизирующий является входным языком оптимизирующего компилятора, который транслирует отлаженные программы.

Язык Ассемблера не имеет собственных средств отладки, но допускает использование общесистемных средств.

Пример 1. Обеспечение дампа SNAP при программировании на языке Ассемблера.

```
//PRIMDUMP JOB  (5Ø8,2—37),'HOCOB',MSGLEVEL=1
//ALL EXEC  ASMFCLG,PARM.ASM='LOAD,NOXREF,NODECK'
//ASM.SYSIN DD *
PRIM1     START
    . . .     . . .
              OPEN (AS,(OUTPUT))
    . . .     . . .
              SNAP DCB=AS,PDATA=(REGS,SAH),STORAGE=(M,M+3)
    . . .     . . .
              RETURN RC=Ø
M             DS    E
AS            DCB DSORG=PS,RECFM=VBA,MACRF=(W),BLKSIZE=882,  *
                  LRECL=125,DDNAME=MYSNAP
    . . .     . . .
              END
/*
//GO.MYSNAP  DD  SYSOUT=A
    . . . . . .
//
```

Дамп SNAP включает трассу областей сохранения (см. п.5.3.1), содержимое общих регистров и регистров плавающей запятой, а также содержимое слова памяти с именем M. Подготовка и печать дампа обеспечиваются макрокомандами OPEN (открывает блок управления данными), SNAP, DCB (формирует блок управления данными) и оператором DD с dd-именем MYSNAP.

Фортран стандартный, являющийся входным языком отладочного компилятора, имеет отладочные операторы, которые дают возможность проверить:

последовательность выполнения операторов любого участка программы — *трассу;*

изменение значений простых переменных и переменных с индексами;

правильность использования индексов;

последовательность обращений к подпрограммам.

Эти возможности обеспечиваются операторами DEBUG, AT, TRACE ON, TRACE OFF и DISPLAY, которые объединяются в пакет отладки.

Пример 2. Применение пакета отладки.

```
      DIMENSION A(20),B(100)
    1 J=5
   10 DO 12 I=1,J
    2 A(I)=I**2
    3 B(I)=I
   12 A(I)=A(I)—B(I)
   15 J=21
   20 F=A(J)
      STOP
      DEBUG TRACE,SUBCHK(A,B)
      AT  1
      TRACE ON
      AT  15
      TRACE OFF
      DISPLAY A
      END
```

Пакет отладки этой программы предусматривает трассировку, которая включается перед оператором с меткой 1 и выключается перед оператором с меткой 15. Каждая метка этого участка программы печатается с предшествующим словом TRACE при каждом выполнении помеченного оператора. Оператор DISPLAY предписывает напечатать значения всех элементов массива A перед выполнением оператора с меткой 15. Режим SUBCHK оператора DEBUG обеспечивает контроль правильности индексов элементов массивов A и B во всей программе. При выполнении данной программы будет выдано сообщение

SUBCHK A(21)

указывающее на ошибку в использовании индекса.

Аналогичные возможности можно использовать в Коболе при помощи глаголов TRACE, EXHIBIT и ON.

В ПЛ/1 для отладки можно воспользоваться оператором ON с отладочными ситуациями CHECK, STRINGRANGE, SUBSCRIPTRANGE и AREA.

Все трансляторы ОС ЕС контролируют синтаксическую правильность исходной программы и выдают диагностические сообщения при обнаружении ошибок. Трансляторы обеспечивают также выдачу диагностических сообщений при выполнении програм-

мы. Для этого в программу при ее трансляции включаются обращения к библиотечным программам обработки ошибок.

Компилятор Алгол-60 подготавливает вырабатываемый объектный модуль к выдаче трассы, если при компиляции указана опция TEST. Трасса представляет собой последовательность номеров знака «точка с запятой» исходного модуля. Трасса выдается, если для пункта выполнения загрузочного модуля указаны параметры TRBEG, TREND, TRACE. Длина трассы определяется комбинацией этих параметров. Параметр DUMP, указанный для пункта выполнения, обеспечивает печать содержимого некоторых участков памяти при появлении ошибок.

Для отладки программ, написанных на языке Ассемблера, имеется специальный отладочный (тестовый) транслятор — Тестран. По указанию программиста Тестран обеспечивает печать информации об изменении содержимого областей памяти и регистров, а также печать трассы.

Помимо перечисленных средств программист может использовать сервисные модули библиотек трансляторов. Вызов нужных модулей предусматривается в исходной программе. В библиотеке Фортрана стандартного, например, имеются сервисные программы SLITE, SLITET, DUMP, PDUMP, DVCHK, OVERFL и EXIT. Программы SLITE, SLITET, DVCHK и OVERFL обеспечивают работу с индикаторами, а DUMP и PDUMP печатают данные. Начало, конец и нужный формат данных указывают в списке параметров оператора CALL, вызывающего программы печати.

Пример 3. Вызов программы печати.

```
CALL PDUMP(A(3),A(10),5,M,M,4)
```

Программа печати PDUMP осуществит вывод вещественных значений третьего — десятого элементов массива A (формат номер 5), а также значения целой переменной M (формат номер 4). Выполнение главной программы (вызывающей) после печати будет продолжено. Отличие программы DUMP состоит в прекращении выполнения главной программы после печати.

Библиотека ПЛ/1 также имеет средства печати содержимого основной памяти. В исходной программе ПЛ/1 для вызова библиотечной программы записывают оператор вида

```
CALL IHEDUMx(аргумент)
```

где на месте x должна быть одна из букв C, J, P или T. Буква определяет вид дампа и режим дальнейшего выполнения программы, как это указано в табл. 5.2. Аргумент представляет собой выражение. Значение выражения появляется в заголовке дампа и может использоваться для нумерации дампов. Значение должно находиться в пределах от 0 до 127, в противном случае оно заменяется на 127. Аргумент надо объявить в программе так:

```
IHEDUMx ENTRY (FIXED BIN(31,0));
```

Но аргумент можно и не указывать. Для получения любого из рассматриваемых дампов необходимо к операторам DD пункта выполнения добавить оператор

```
//PL1DUMP  DD  SYSOUT=A
```

Таблица 5.2

Разновидности дампов

Буква	Характеристика дампа
C	Печать содержимого основной памяти текущей задачи с последующим продолжением выполнения задачи
J	Печать содержимого основной памяти всех активных задач с последующим продолжением их выполнения
P	Печать содержимого основной памяти всех активных задач с последующим завершением старшей задачи
T	Печать содержимого основной памяти текущей задачи с последующим завершением этой задачи

5.2. ОВЕРЛЕЙНЫЕ СТРУКТУРЫ

5.2.1. Планирование

Редактор связей вырабатывает загрузочный модуль простой или оверлейной структуры (структуры с перекрытием).

В процессе исполнения модуля простой структуры все его программные секции одновременно находятся в основной памяти (рис. 5.1). Объем необходимой памяти определяется суммой длин всех секций модуля.

Загрузочный модуль оверлейной структуры состоит из сегментов (рис. 5.2). Сегмент содержит одну или несколько секций и представляет собой наименьшую часть модуля, которая во время выполнения может загружаться в память как единое логическое целое. При выполнении загрузочного модуля оверлейной структуры один сегмент, называемый корневым, постоянно находится в основной памяти (сегмент A на рис. 5.2). Другие сегменты загружаются в память по мере необходимости. При этом загрузка некоторых сегментов производится с перекрытием, т. е. загружа-

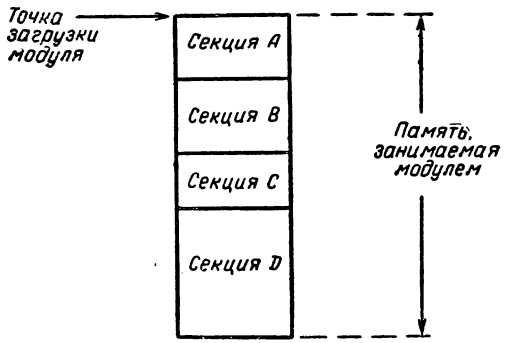


Рис. 5.1. Загрузочный модуль простой структуры

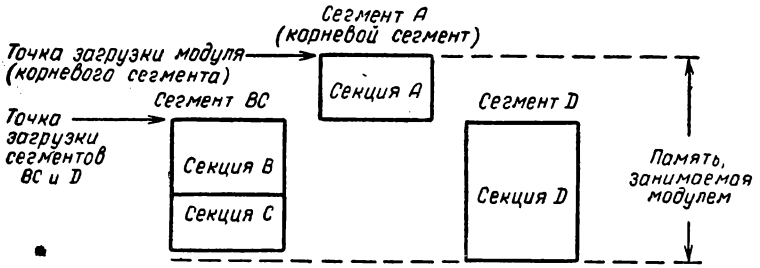


Рис. 5.2. Загрузочный модуль оверлейной структуры

емый сегмент располагается на месте одного или нескольких сегментов, загруженных ранее и завершивших свое функционирование. Взаимно перекрывающиеся сегменты BC и D (см. рис. 5.2) загружаются в память начиная с одного и того же адреса. Загрузка производится столько раз, сколько этого требует логика программы. В рассматриваемом примере необходимая память определяется суммой длин сегментов A и D. Кроме того, следует выделить память для размещения оверлейного супервизора и таблиц, обеспечивающих оверлейный процесс (см. п. 5.2.4).

Использование загрузочных модулей простой структуры более эффективно по времени выполнения. Однако, если размер программы превышает объем выделенной для нее основной памяти, приходится использовать оверлейную структуру. В этом случае программист должен спланировать структуру оверлейного модуля и описать ее редактору связей при помощи управляющих предложений. Методика планирования в значительной степени зависит от связей между секциями программы.

Пример. Пусть требуется спланировать оверлейную структуру программы, состоящей из секций A, B, C, D, E, F и G (рис. 5.3). Между секциями существуют связи, показанные на рисунке линиями (направление связей несущественно). Выполнение программы должно начинаться с секции A.

Методика планирования базируется на использовании графического представления структуры программы в виде дерева. Каждой секции программы соответствует узел дерева. Самым верхним узлом (корнем) изображается секция, выполняемая первой. Ветви дерева получаются в результате последовательного просмотра

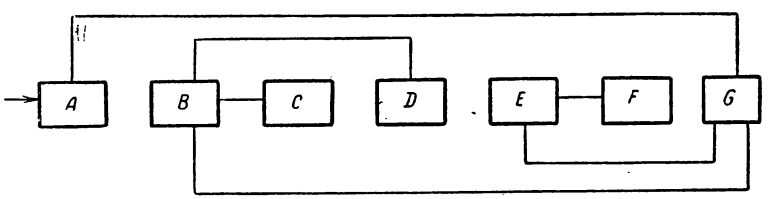


Рис. 5.3. Связи секций программы

всех связей программы. При этом наличие трех и более связей у одной секции порождает разветвление дерева. На рис. 5.4, *а* показано дерево для рассматриваемой программы. Три пути от корня до листьев дерева соответствуют трем цепочкам связанных секций (A—G—B—C, A—G—B—D, A—G—E—F). Секция, входящая в несколько цепочек (любая из секций A, B или G), является общей для неповторяющихся участков этих цепочек. Секции, общие для одних и тех же участков (секции A и G), объединяются в сегменты. Сегмент, содержащий секции, общие для всех цепочек, будет корневым. В сегмент объединяется также участок цепочки, не содержащий общих секций (участок с секциями E и F). После таких преобразований каждый узел дерева соответствует сегменту программы (рис. 5.4, *б*). Это дерево отражает оверлейную структуру программы. Здесь сегмент AG — корневой. Сегменты C и D не имеют взаимных связей, и, следовательно, одновременное присутствие их в памяти не требуется. Для них нужно назначить один и тот же адрес загрузки. Во время выполнения программы они будут перекрывать друг друга при загрузке. То же самое относится к паре сегментов B и EF. Загрузка перекрывающихся сегментов и передача им управления обеспечиваются их общим сегментом, поэтому в момент загрузки некоторого сегмента все предшествующие ему в цепочке сегменты должны находиться в основной памяти.

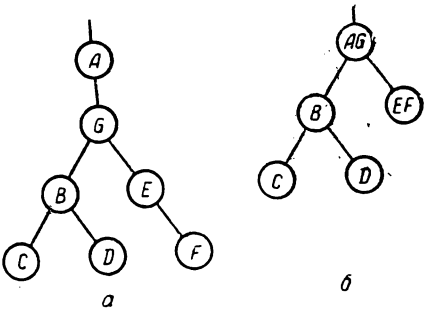


Рис. 5.4. Дерево структуры программы

При некоторых вариантах связей между секциями программы применение рассмотренной методики планирования оказывается невозможным (см. п. 5.2.4).

Загрузочные модули оверлейной структуры могут занимать одну или несколько областей основной памяти. Область — это непрерывный участок памяти. Однообластная оверлейная структура использует один такой участок, а многообластная — несколько (до четырех).

Необходимость применения многообластной структуры может возникнуть по следующей причине. Если оверлейная структура содержит одинаковые секции (например, секции D или E на рис. 5.5,*а*), то редактор связей обработает только первую из них, игнорируя остальные. Преодолеть эту трудность можно, включив повторяющиеся секции в корневой сегмент. Однако в таком случае в корневом сегменте могут оказаться секции, которые взаимно не связаны и могли бы перекрывать друг друга (секции D и E на рис. 5.5, *б*). Чтобы использовать возможность перекрытия, применяют многообластную структуру, в каждой области которой действуют правила однообластной структуры. На рис. 5.5, *в* представ-

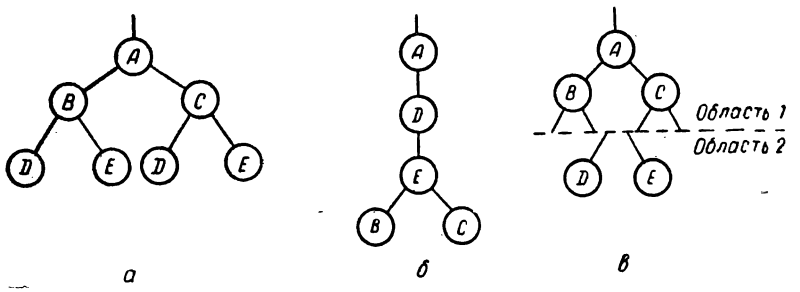


Рис. 5.5. Однообластная и многообластная структуры

лена оверлейная структура, использующая две области. Область 2 служит для размещения перекрывающихся секций D и E. Секция D (или E) загружается в область 2 при обращении к ней из области 1, независимо от того, какая из секций (B или C) находится в области 1.

5.2.2. Описание

Для получения загрузочного модуля запланированной оверлейной структуры в задании нужно указать:

оверлейный режим работы редактора связей;

расположение сегментов оверлейной структуры;

порядок включения секций в сегменты.

Оверлейный режим редактора связей указывают посредством опции OVLY в управляющем предложении EXEC для шага редактирования, например

```
//LKED EXEC PGM=IEWL,PARM='OVLY'
```

Расположение сегментов оверлейной структуры в основной памяти описывают последовательностью управляющих предложений OVERLAY редактора связей. Каждому сегменту структуры, кроме корневого, в этой последовательности соответствует одно такое предложение. Порядок следования предложений соответствует порядку обхода дерева оверлейной структуры сверху вниз, слева направо и область за областью. Предложение OVERLAY записывается по обычным правилам (см. п. 4.2.6) в одном из двух форматов:

```
OVERLAY имя
```

или

```
OVERLAY имя(REGION)
```

Здесь «имя» идентифицирует точку загрузки (начальный адрес) сегмента. Это имя можно выбрать произвольно, но для перекрывающихся сегментов, имеющих одну и ту же точку загрузки, обязательно используют одно и то же имя. Указатель REGION записывают только в предложении OVERLAY для первого сегмента

каждой области (кроме первой). Таким образом, предложение OVERLAY служит для указания начала сегмента и начала области.

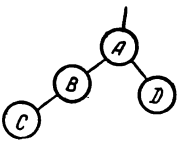


Рис. 5.6. Дерево оверлейной структуры

Вслед за каждым предложением OVERLAY в задании указывают секции, включаемые в соответствующий сегмент. Это можно сделать тремя способами:

непосредственным размещением карт объектных модулей, содержащих секции сегмента;

при помощи управляющих предложений INCLUDE редактора связей;

посредством управляющих предложений INSERT редактора связей.

При первом способе объектные модули помещают во входной поток, связанный с набором данных основного ввода редактора.

Пример 1. Описание оверлейной структуры, дерево которой показано на рис. 5.6.

Объектные модули, содержащие секции A, B, C и D, находятся во входном потоке. Каждый модуль состоит из одной секции.

```
//LKED EXEC PGM=IEWL,PARM='OVLY'
. . . . . .
//SYSLIN DD *
      ENTRY VHOD
            | Объектный модуль, состоящий из секции A |
      OVERLAY PEREKR
            | Объектный модуль, состоящий из секции B |
            | Объектный модуль, состоящий из секции C |
      OVERLAY PEREKR
            | Объектный модуль, состоящий из секции D |
/*
```

Предложение INCLUDE (см. п. 4.2.6) включает в сегмент объектные модули из набора данных дополнительного ввода для редактора.

Пример 2. Описание структуры, рассмотренной в предыдущем примере.

Объектные модули, включающие в себя секции A, B, C и D, находятся в библиотечном наборе данных MODBIB. Каждый модуль содержит одну секцию. Имена членов библиотеки совпадают с именами секций.

```
//LKED EXEC PGM=IEWL,PARM='OVLY'
. . . . . .
//MODBIB DD DSNAME=OBJBIB,DISP=(OLD,KEEP),...
. . . . . .
//SYSLIN DD *
  ENTRY VHOD
  INCLUDE MODBIB(A)
  OVERLAY PEREKR
  INCLUDE MODBIB(B,C)
```

```
 OVERLAY PEREKR
 INCLUDE MODBIB(D)
/*
```

Предложение INSERT обеспечивает перемещение секций с их места в наборе данных основного ввода в нужный сегмент. Предложение имеет вид:

INSERT список имен секций

Имена в списке разделяются запятыми и идентифицируют секции, подлежащие перемещению.

Пример 3. Описание структуры, показанной на рис. 5.6.

Объектные модули, содержащие секции A, B, C и D, находятся в наборе данных основного ввода редактора связей. Секции могут быть в разных объектных модулях, как в примерах 1 и 2, или могут входить в один объектный модуль.

```
//LKED EXEC PGM=IEWL,PARM='OVLY'
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
//SYSLIN DD DSNAME=OBJBIB,DISP=(OLD,KEEP),...
//       DD  *
 ENTRY VHOD
 INSERT A
 OVERLAY PEREKR
 INSERT B,C
 OVERLAY PEREKR
 INSERT D
/*
```

Предложение INSERT используют также для пересылки в нужный сегмент автоматически вызываемых секций, которые первоначально всегда включаются в корневой сегмент.

Можно применять различные комбинации рассмотренных способов указания секций. Во избежание ошибок имя входа в корневой сегмент рекомендуется указывать всегда. Это делают при помощи управляющего предложения ENTRY редактора связей (см. п. 4.2.6). Имя входа должно быть обязательно определено в программной секции корневого сегмента как внешнее имя. В предыдущих примерах указывалась точка входа VHOD, определенная в секции A.

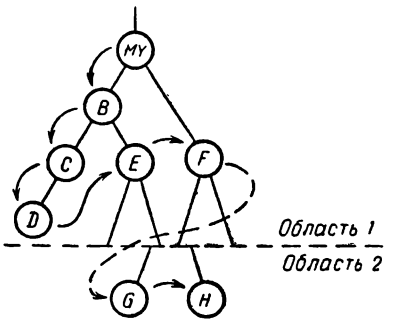


Рис. 5.7. Дерево оверлейной структуры с двумя областями

Пример 4. Задание на формирование и исполнение оверлейной программы.

Исходная программа написана на Фортране и состоит из главной программы MY (имя указано в операторе EXEC) и подпрограмм B, C, D, E, F, G и H. В результате пакетной компиляции создается восемь объектных модулей, каждый из которых содержит одну программную секцию. Порядок обхода узлов дерева (секций) при описании структуры показан на рис. 5.7 стрелками. Необходимые обслуживающие секции Фортрана вызываются автоматически из библиотеки SYS1.FORTLIB на этапе редактирования и поэтому поме-

щаются в корневой сегмент. Задание использует каталогизированную процедуру компиляции, редактирования и выполнения.

```
//OVERSTR JOB (500,2—37),'HOCOB',MSGLEVEL=1
//ALL EXEC FORTGCLG,PARM.FORT='NOLIST,NOMAP,NAME=MY',
//                  PARM.LKED='OVLY,LIST,XREF'
//FORT.SYSIN DD *
```

Исходные модули Фортрана

```
/*
//LKED.SYSIN DD *
 ENTRY MY
 INSERT MY
 OVERLAY BF
 INSERT B
 OVERLAY CE
 INSERT C,D
 OVERLAY CE
 INSERT E
 OVERLAY BF
 INSERT F
 OVERLAY GH(REGION)
 INSERT G
 OVERLAY GH
 INSERT H
/*
//GO.SYSIN DD *
```

Исходные данные

```
/*
//
```

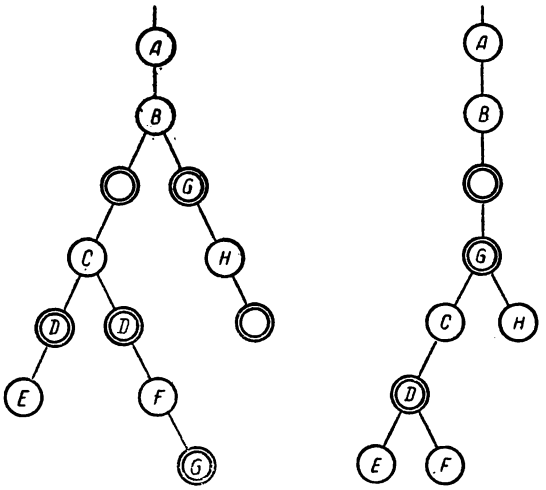
5.2.3. Некоторые особенности

Общие области. Редактор связей автоматически помещает общую область сегментов различных цепочек (ветвей дерева) в общий сегмент этих цепочек. При этом из одноименных или непоименованных областей выбирается наибольшая. Поименованная общая область может быть помещена в нужный сегмент оверлейной структуры при помощи управляющего предложения INSERT.

На рис. 5.8, *а* показан пример дерева оверлейной структуры перед обработкой общих областей редактором связей. Узлы дерева, обозначенные двойными линиями, соответствуют поименованным и непоименованным общим областям. На рис. 5.8, *б* изображено дерево этой же оверлейной структуры после обработки общих областей.

Оценка памяти. При выполнении оверлейной программы должна быть выделена основная память для загрузочного модуля, оверлейного супервизора и списка примечаний.

Память, необходимая для размещения оверлейного загрузочного модуля, определяется суммарной длиной всех областей структуры. Области отводятся последовательно с выравниванием на границу двойного слова. Длина области равна длине наиболее протяженной цепочки связанных сегментов. Некоторые сегменты содер-



а *б*

Рис. 5.8. Дерево оверлейной структуры с общими областями

жат оверлейные таблицы, формируемые редактором связей. В этих таблицах записана информация, необходимая для работы оверлейного супервизора. Таблицы бывают двух видов: таблицы сегментов (SEGTAB) и таблицы входов (ENTAB). Каждая таблица оформляется как секция оверлейной программы. На рис. 5.9 показано расположение оверлейных таблиц в структуре загрузочного модуля из примера 4 (см. п. 5.2.2). Символом «*» на рисунке условно помечены все секции, включенные автоматически из библиотеки SYS1. FORTLIB.

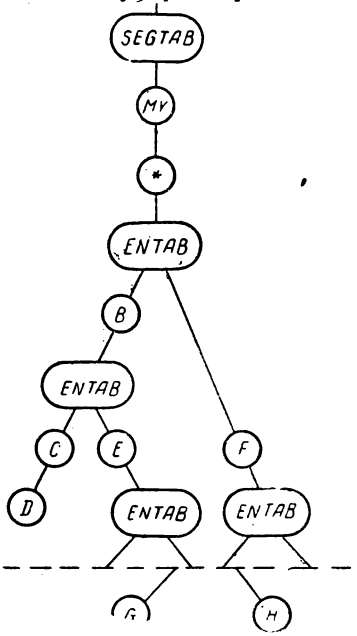


Рис. 5.9. Таблицы оверлейной программы

Любая оверлейная программа содержит только одну таблицу сегментов, причем она является первой секцией корневого сегмента. В этой таблице записаны сведения о сегментах и областях программы. Во время выполнения программы в таблице отмечается, какие сегменты находятся в памяти и какие подлежат загрузке.

Таблица входов — последняя

секция сегмента. В ней формируется элемент для каждой ссылки из данного сегмента к сегменту, расположенному ниже по цепочке (дальше от корневого сегмента). Элемент для ссылки не создается в таблице данного сегмента, если аналогичный элемент уже есть в таблице входов сегмента, расположенного ближе к корневому.

Ниже указаны размеры областей памяти (в байтах), необходимые для программных средств, обеспечивающих оверлейный процесс.

Оверлейный супервизор (максимальный вариант)	992
Таблица сегментов SEGTAB	4n+24
Таблица входов ENTAB	12(x+1)
Список примечаний NOTELST	4n+8

Здесь n — число сегментов в оверлейной структуре;
x — число элементов в таблице входов.

Исключающие ссылки. В некоторых случаях планирование оверлейной структуры по методике, изложенной в п. 5.2.1, оказывается невозможным, например, когда программа имеет только одну цепочку связанных секций (рис. 5.10, *а*). Рассмотрим особенности планирования оверлейной структуры такой программы. Введем для этого понятие путь, определив его как последовательность секций (сегментов), которые могут находиться в основной памяти одновременно. Ясно, что оверлейная структура должна содержать по меньшей мере два пути. Ссылки между сегментами одного пути называются *включающими*. Программист может поместить связанные секции в разные пути (секции B и C в путях A — B и A — C на рис. 5.10, *б*). При этом появляются *исключающие* ссылки — ссылки между сегментами различных путей одной области. Ссылки могут быть правильные и неправильные. Включающие ссылки всегда правильные. Исключающая ссылка называется правильной, если к одному и тому же символу вызываемого сегмента имеется как исключающая ссылка, так и включающая ссылка из общего сегмента. В противном случае исключающая ссылка называется неправильной. На рис. 5.10, *б* включающая ссылка из секции A к секции C отсутствует (показано прерывистой линией). Поэтому ссылка из секции B к секции C — неправильная исключающая ссылка. Правильной может быть только ссылка из секции C к секции B (показана линией со стрелкой).

При правильной исключающей ссылке таблица ENTAB общего сегмента содержит элемент для символа, на который делается

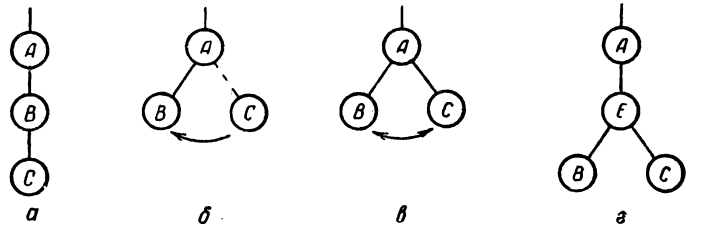


Рис. 5.10. Включающие и исключающие ссылки

ссылка. Таблица входов вызывающего сегмента такой элемент не включает в себя (как расположенная ниже по пути). В этом случае загрузка и передача управления сегменту, перекрывающему вызывающий, осуществляется при помощи таблицы ENTAB общего сегмента.

Загрузочный модуль, содержащий правильные или неправильные исключающие ссылки, отмечается редактором связей как НЕВЫПОЛНИМЫЙ. Этот же модуль будет ВЫПОЛНИМЫМ, если записать опцию LET в управляющем предложении EXEC для пункта редактирования, например

```
//REDAKT EXEC PGM=IEWL,PARM='OVLY,LET'
```

Указанная возможность может быть полезна при отладке большой программы по частям. Опцию XCAL редактора связей можно использовать для получения ВЫПОЛНИМОГО модуля в том случае, если все исключающие ссылки между сегментами модуля — правильные.

Выполнение неправильной исключающей ссылки в оверлейной программе, отредактированной с опцией LET, приводит к неверному результату. Такие ссылки должны быть устранены. Это можно сделать по крайней мере двумя способами.

Первый способ состоит в преобразовании неправильной ссылки в правильную путем размещения необходимой включающей ссылки в общем сегменте (см. рис. 5.10, *в*). При этом включающая ссылка может быть даже фиктивной (невыполняемой), поскольку она необходима только для создания элемента таблицы ENTAB.

Второй способ заключается во введении в общий сегмент дополнительной секции (секция E на рис. 5.10, *г*), через которую организуется связь между перекрывающими друг друга сегментами. В этом случае исключающие ссылки полностью устраняются.

5.3. МЕЖМОДУЛЬНЫЕ СВЯЗИ

5.3.1. Соглашения о связях. Область сохранения

Возможны два подхода к организации связей между объектными модулями. Один из подходов предполагает раздельное редактирование модулей и выполнение полученных загрузочных модулей в различных пунктах задания. Связь между такими модулями может осуществляться через общие наборы данных. При другом подходе объектные модули объединяются редактором связей в загрузочный модуль. Ниже рассматривается организация связей между модулями, выполняемыми в составе одного загрузочного модуля (одной программы).

Если один модуль программы передает управление другому, то первый из них называют вызывающим, а второй — вызываемым (или подпрограммой). По отношению к различным модулям программы один и тот же модуль может быть как вызывающим, так и вызываемым.

Вследствие независимой трансляции разные модули используют одни и те же регистры, поэтому при переходе от одного модуля к другому приходится сохранять содержимое регистров вызывающего модуля в специальной *области сохранения.* Каждый объектный модуль имеет свою область сохранения, которая либо назначается при трансляции, либо выделяется динамически в ходе выполнения программы. Например, компилятор Фортрана предусматривает область сохранения внутри объектного модуля, а компилятор ПЛ/1 формирует в объектном модуле команды вызова обслуживающего модуля, который резервирует область сохранения при выполнении программы.

Информация, помещенная в область сохранения вызывающего модуля, используется перед возвратом из вызываемого модуля для восстановления содержимого регистров.

Различают две формы межмодульных связей: связь по управлению и связь по данным. *Связь по управлению* обеспечивает передачу и возврат управления, а также сохранение и восстановление регистров, а *связь по данным* — передачу аргументов (параметров) из модуля в модуль.

При формировании команд, организующих межмодульные связи, компиляторы выполняют стандартные *соглашения о связях.* Эти соглашения устанавливают правила использования регистров общего назначения для организации связи и определяют стандартную структуру области сохранения. Загрузку регистров связи производит вызывающий модуль. Использование регистров, не указанных в табл. 5.3, соглашения о связях не регламентируют.

Таблица 5.3

Регистры связи

Регистр	Название	Содержимое регистра
1	Регистр списка параметров	Адрес списка параметров, передаваемых вызываемому модулю
13	Регистр области сохранения	Адрес области сохранения вызывающего модуля
14	Регистр возврата	Адрес возврата в вызывающий модуль
15	Регистр точки входа	Адрес входа в вызываемый модуль

Стандартная область сохранения занимает 18 слов (72 байта). Назначение слов показано в табл. 5.4. В этой таблице СОХР — символическое имя начала области сохранения. Перед выходом в вызываемый модуль адрес области сохранения загружается в регистр 13. После входа в вызываемый модуль регистр 13 используется для того, чтобы переписать содержимое регистров вызвавшего модуля в его область сохранения.

Таблица 5.4

Область сохранения модуля

Адрес	Слово	Содержимое слова
СОХР	1	Адрес эпилога программы на Фортране Индикаторы, программы на ПЛ/1
СОХР+4	2	Адрес области сохранения модуля, вызвавшего данный модуль
СОХР+8	3	Адрес области сохранения вызываемого модуля
СОХР+12	4	Содержимое регистра 14
СОХР+16	5	Содержимое регистра 15
СОХР+20	6	Содержимое регистра 0
СОХР+24	7	Содержимое регистра 1
.	.	. . . . .
СОХР+64	17	Содержимое регистра 11
СОХР+68	18	Содержимое регистра 12

5.3.2. Связь по управлению

Вызванный программистом модуль возвращает управление вызвавшему модулю, хотя это не обязательно. Пример такой простейшей связи показан на рис. 5.11. Модуль A получает управление от управляющей программы и является вызывающим для модуля B. В свою очередь модуль B вызывает модуль C, а модуль C — модуль D. Модули A, B и C имеют собственные используемые области сохранения. Модуль D не является вызывающим, поэтому его область сохранения не используется (хотя она может быть выделена).

Организацию связей по управлению рассмотрим на примере связи модулей B и C (см. рис. 5.11). Действия, обеспечивающие связь, описаны ниже.

1. Действия, выполняемые в модуле B при вызове модуля C:

а) выделение области сохранения B с выравниванием ее нача-

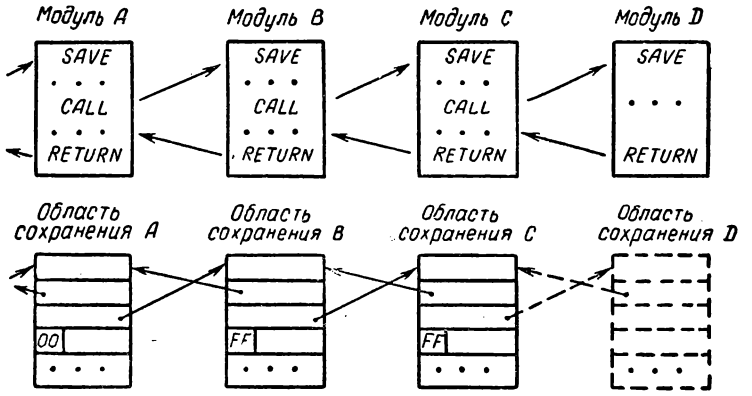


Рис. 5.11. Связь по управлению

ла на границу слова (это действие может выполняться заранее, например, при входе в модуль B или при его компиляции);

б) загрузка регистра 13 адресом области сохранения B (с предварительной перегрузкой в другой регистр адреса области сохранения A);

в) запись адреса области сохранения B в третье слово области сохранения A (при динамическом выделении области сохранения это выполняется в модуле A);

г) запись адреса области сохранения A во второе слово области сохранения B;

д) загрузка регистра 15 адресом входа в модуль C;

е) загрузка регистра 14 адресом возврата в модуль B;

ж) передача управления по адресу, указанному в регистре 15.

2. Действия, выполняемые при входе в модуле C:

а) запись содержимого регистров модуля B в слова 4—18 области сохранения B;

б) организация базирования в модуле C.

3. Действия, выполняемые в модуле C при возврате в модуль B:

а) загрузка регистра 13 адресом области сохранения B;

б) восстановление содержимого регистров 14,15,0—12 модуля B загрузкой в них информации, хранящейся в словах 4—18 области сохранения B;

в) запись в старший байт четвертого слова области сохранения B признака ее освобождения FF. Неосвобожденные области имеют признак 00;

г) передача управления (возврат) по адресу, указанному в регистре 14.

Признак состояния области сохранения (FF или 00) облегчает анализ дампов и трасс областей сохранения при отладке программ. Признакам состояния областей сохранения, показанным на рис. 5.11, соответствует выполнение модуля B после возврата управления от модуля C.

Действия, аналогичные описанным выше, выполняются при организации связи по управлению между любой парой модулей. Программирует эти действия компилятор, а для программ на языке Ассемблера — сам программист. Для упрощения программирования межмодульных связей на языке Ассемблера можно использовать макрокоманды CALL, SAVE и RETURN.

Макрокоманда

```
CALL C
```

используемая без параметров, обеспечивает переход к модулю с именем C с последующим возвратом к предложению, следующему в программе за этой макрокомандой, т. е. обеспечивает действия *д — ж* в вызывающем модуле.

Макрокоманда SAVE используется для выполнения действия *а* при входе в модуль. Она позволяет сохранить содержимое нужной группы регистров вызвавшего модуля. Например, макрокоманда

```
SAVE (14, 12)
```

обеспечивает запись в область сохранения содержимого регистров 14, 15,0—12. Если надо сохранить содержимое одного регистра, в качестве параметра указывают номер только этого регистра. Параметр T в макрокоманде SAVE добавляет к этому регистру регистры 14 и 15. Так, макрокоманда

```
SAVE (2),T
```

предписывает сохранить содержимое регистров 2, 14 и 15.

Макрокоманда SAVE, кроме того, дает возможность идентифицировать программную секцию, в которой эта макрокоманда записана. Например, макрорасширение макрокоманды

```
SAVE (14,3),*
```

будет содержать в пятом байте указатель длины имени программной секции, а вслед за указателем — собственно имя. Это обеспечивается параметром «*». Если макрокоманда помечена, тогда в этих байтах размещается указатель длины метки и метка. Программная секция может быть идентифицирована любым другим именем. Для этого в макрокоманде вместо звездочки нужно записать любой идентификатор, содержащий не более 70 символов. Этот идентификатор будет включен в макрорасширение вместо имени программной секции. Описанная возможность повышает удобство работы с дампами при отладке программ.

Макрокоманда RETURN применяется для выполнения действий *б — г* при выходе из модуля, она позволяет восстановить содержимое одного регистра или группы регистров. Например, макрокоманда

```
RETURN (14, 12)
```

восстанавливает содержимое регистров 14,15,0—12 и возвращает управление вызвавшему модулю. Для восстановления содержимого одного регистра его номер указывают в качестве параметра макрокоманды.

Параметр T в макрокоманде RETURN предписывает пометить область сохранения вызвавшего модуля признаком FF (свободная).

Макрокоманда RETURN может содержать параметр, устанавливающий код возврата, например,

```
RETURN(14,8),T,RC=12
```

Код возврата — число, кратное 4, из диапазона 0—4095. В качестве кода нормального возврата в системе принят 0. Анализ кода возврата должен быть предусмотрен в вызывающем модуле. Код возврата помещается в регистр 15 при выполнении макрокоманды. В данном случае вызвавшему модулю будет передан код возврата 12.

Пример 1. Возможная структура модуля B, написанного на языке Ассемблера.

```
                                                    Номер
                                                    предложения
B       START   Ø                                       1
        SAVE    (14,12),*                               2
        BALR    2,Ø                                     3
        USING   *,2                                     4
        LR      4,13                                    5
        LA      13,SOHR                                 6
        ST      13,8(Ø,4)                               7
        ST      4,4(Ø,13)                               8
        . . . . . .
        CALL    C                                       9
        . . . . . .
        L       13,SOHR+4                              10
        RETURN  (14,12),T,RC=Ø                         11
SOHR    DS      18F                                    12
        END                                            13
```

Модуль B вызывает модуль C. Выполняемые при этом действия описаны предложениями 5—9, 12. С другой стороны, модуль B сам является вызываемым. Предложения 2—4 предписывают выполнить необходимые действия при входе в модуль B, а предложения 10—11 — при возврате из него.

Заметим, что многие системные макрокоманды помимо регистров связи используют также регистр 0. Обычно в таких макрокомандах регистр 0 используется для передачи параметров. Это нужно учитывать при включении макрокоманд в программу на языке Ассемблера.

5.3.3. Связь по данным

Организация связи по данным между вызывающим и вызываемым модулями обеспечивает передачу параметров вызываемому модулю и возврат результата, если это необходимо, вызвавшему модулю.

В вызывающем модуле для передачи параметров нужно: а) сформировать список параметров; б) загрузить регистр 1 адресом начала списка параметров.

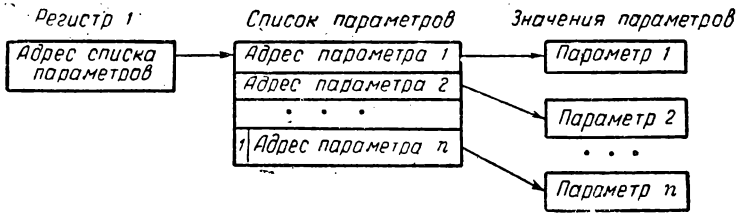


Рис. 5.12. Передача параметров

Список параметров выравнивается на границу слова памяти и состоит из элементов длиной четыре байта каждый. Элемент списка содержит адрес, который в простейшем случае является адресом значения параметра. Последний элемент имеет единицу в левом бите слова — признак конца списка параметров. Адрес списка параметров передается в вызываемый модуль в регистре 1 (рис. 5.12).

При программировании на языке Ассемблера передачу параметров можно описать либо средствами языка, либо при помощи макрокоманды CALL.

Пример 1. Макрокоманда вызова с передачей параметров.

```
CALL C,(J,Y,X),VL
```

Макрокоманда передает управление модулю C, обеспечивает формирование списка параметров J, Y, X и загрузку регистра 1. Параметр VL устанавливает признак конца списка.

Извлечение параметров в вызываемом модуле на языке Ассемблера описывается средствами языка.

Пример 2. Извлечение значения параметра.

```
L 3,8(Ø,1)
LE 2,Ø(Ø,3)
```

Предложения языка Ассемблера предписывают извлечь параметр X в модуле C (см. пример 1) и записать его значение в регистр 2 плавающей запятой.

При передаче параметров нужно следить, чтобы атрибуты параметров (тип, длина, выравнивание и т. п.) в вызывающем и вызываемом модулях были одинаковы.

Возврат результатов из вызванного модуля в вызвавший обычно обеспечивается соответствующими параметрами в списке параметров. В некоторых случаях используются специальные дополнения к стандартным соглашениям о связях. Например, подпрограмма-функция Фортрана возвращает значение вещественной функции или функции двойной точности в регистре 0 плавающей запятой, а значение целой функции — в общем регистре 0.

Связь по данным можно организовать также при помощи общих областей памяти (см., например, листинги на рис. 4.9 и рис. 4.10).

5.3.4. Особенности связи модулей, транслированных с разных языков

Организация связи модулей, написанных на одном языке, обычно не вызывает трудностей. Компиляторы обеспечивают связь по управлению и данным автоматически.

Имеются некоторые особенности в организации связи модулей РПГ. Соглашения о связях здесь выполняются только в отношении регистров 14 и 15, причем вызываемый модуль не обеспечивает возврата в вызывающий. Передача данных осуществляется при помощи операций RLABL и ULABL.

При программировании на языке Ассемблера рекомендуется придерживаться стандартных соглашений о связях, однако опытный программист может организовать связь по своему усмотрению.

На связь модулей, транслированных с разных языков, влияют особенности языковых средств и компиляторов. Языковые средства различаются неэквивалентными типами данных. К особенностям компиляторов относятся неполное или специфичное выполнение соглашений о связях, а также несходство в организации хранения данных эквивалентных типов. Названные различия можно учесть в специальном промежуточном модуле, написанном на языке Ассемблера. Некоторые возможности непосредственной связи модулей, полученных трансляцией с различных языков высокого уровня, рассматриваются ниже.

Компилятор Алгола-60 создает особенности в организации передачи параметров, накладывает ограничения на использование регистров и требует особого определения точки входа в вызываемую программу. По этим причинам непосредственная связь модулей Алгола-60 с модулями других языков высокого уровня затруднительна. При учете отмеченных выше особенностей модули Алгола-60 можно связывать с модулями языка Ассемблера.

Модули РПГ можно связывать с модулями других языков. При этом модуль РПГ может быть как вызывающим, так и вызываемым. Поскольку вызывающий модуль РПГ не выделяет область сохранения, ее нужно предусматривать в вызываемом модуле. Отсутствие в некоторых языках десятичных упакованных данных, с которыми оперируют модули РПГ, делает прямую связь с модулями этих языков по данным невозможной.

Компиляторы Фортрана и Кобола выполняют стандартные соглашения о связях одинаковым образом [10]. В вызывающих объектных модулях компиляторы формируют, в частности, команды, соответствующие макрорасширению системной макрокоманды CALL, рассмотренной выше. Передача параметров производится согласно схеме, показанной на рис. 5.12. Это позволяет связывать модули Фортрана и Кобола. Напомним, что в Коболе отсутствуют языковые средства для работы с функциями (указатели функций), поэтому модуль Кобола может вызывать только подпрограмму Фортрана. Необходимо, кроме того, учитывать различия в типах данных и особенности хранения данных в памяти.

Значительные трудности возникают при связывании модулей ПЛ/1 с модулями других языков. Эти трудности проявляются как при передаче данных, так и при передаче управления. Рассмотрим особенности связи, которые необходимо учитывать при написании программ и составлении заданий.

Связь по данным модуля ПЛ/1 с модулями других языков имеет особенности, обусловленные неэквивалентностью типов данных, способов хранения и передачи данных.

Таблица 5.5

Соответствие данных ПЛ/1 и Фортрана

ПЛ/1				Фортран		
		Выравнивание				
Тип	Длина, байт	ALIGNED	UNALIGNED	Тип	Длина, байт	Выравнивание
REAL FIXED BIN	2	Полуслово	Байт	INTEGER	2	Полуслово
» » »	4	Слово	»	»	4	Слово
REAL FLOAT BIN	4	»	»	—	—	—
» » »	8	Двойное слово	»	—	—	—
REAL FIXED DEC	~	Байт	»	—	—	—
REAL FLOAT DEC	4	Слово	»	REAL	4	Слово
» » »	8	Двойное слово	»	»	8	Двойное слово
CPLX FLOAT DEC	8	Слово	»	COMPLEX	8	Слово
» » »	16	Двойное слово	»	»	16	Двойное слово
BIT	~	Байт	Бит	—	—	—
»	1	»	»	LOGICAL	1	Байт
»	4	»	»	»	4	»
CHAR	~	»	Байт	—	—	—

Примечание. Знак ~ означает здесь и далее произвольную допустимую длину.

ПЛ/1 имеет значительно больше типов данных, чем Фортран. Соответствие некоторых типов данных ПЛ/1 и Фортрана показано в табл. 5.5. При распределении памяти компилятор Фортрана всегда производит выравнивание данных, соответствующее выравниванию данных ПЛ/1 с атрибутом ALIGNED, а данные ПЛ/1 могут быть объявлены как выравниваемые (атрибут ALIGNED) или невыравниваемые (атрибут UNALIGNED). Поэтому данные ПЛ/1, которые ставятся в соответствие данным Фортрана, нужно объявлять ALIGNED. Однако, если данные Фортрана объединяются в неформатную запись, то выравнивание теряется, и каждый элемент записи Фортрана размещается на границе байта. Соответствующая структура ПЛ/1 должна иметь атрибут UNALIGNED. При объеди-

нении данных Фортрана в неформатную запись можно сделать искусственное выравнивание путем размещения элементов списка вывода в порядке уменьшения длин, соответствующих типам этих элементов.

Пример 1. Искусственное выравнивание в неформатной записи.

```
INTEGER*2 A
LOGICAL*1 B
REAL*8 C
. . . . . .
WRITE(6) C,K,A,B
```

В списке неформатного вывода указаны переменные C (длина 8 байт), K (длина 4 байта), A (длина 2 байта) и B (длина 1 байт). Такой порядок следования переменных обеспечивает выравнивание данных. Соответствующую структуру ПЛ/1 нужно объявить ALIGNED.

Многомерные массивы ПЛ/1 и Фортрана хранятся в памяти по-разному. В Фортране массив отображается в вектор по столбцам, а в ПЛ/1 — по строкам. Согласование таких массивов можно обеспечить либо описанием массивов в разных языках с противоположным порядком следования индексов, либо использованием iSUB-определения.

Пример 2. Применение противоположной индексации.

Модуль Фортрана	Модуль ПЛ/1
INTEGER TAB(2,3,6)	DCL TAB(6,3,2) FIXED BIN(31);
. . . . .	. . . . .

Порядок размещения элементов массивов TAB Фортрана и TAB ПЛ/1 согласован. Противоположная индексация неудобна тем, что один и тот же элемент массива в разных модулях имеет разные индексы. Например, элементу TAB (1,2,4) в Фортране соответствует элемент TAB (4,2,1) в ПЛ/1.

Пример 3. Применение iSUB-определения.

Модуль Фортрана	Модуль ПЛ/1
. . . . .	. . . . .
INTEGER TAB(2,3,6)	DCL TAB1(6,3,2) FIXED BIN(31),
. . . . .	TAB(2,3,6) FIXED BIN(31) DEF
	TAB1(3SUB,2SUB,1SUB);
	. . . . .

Порядок размещения элементов массивов TAB Фортрана и TAB1 ПЛ/1 согласован. Массив TAB ПЛ/1 по определению есть массив TAB1 с переименованными индексами элементов. Таким образом, использованием массива TAB в модуле ПЛ/1 достигается согласование нумерации элементов массивов разных модулей. Так, элемент TAB (1, 2, 4) имеет одинаковый порядковый номер в массивах обоих модулей.

В ПЛ/1 и Коболе типы данных и их организация во многом одинаковы (табл. 5.6). Выравнивание данных ПЛ/1 с атрибутом ALIGNED соответствует выравниванию данных SYNCHRONIZED Кобола. Невыравниваемые данные ПЛ/1 (UNALIGNED) и Кобола (UNSYNCHRONIZED) размещаются на границе байта. Однако способы хранения структур ПЛ/1 и записей Кобола могут приводить к их несоответствию:

компилятор ПЛ/1 минимизирует объем неиспользуемой памяти в структуре. Достичь согласования можно путем использования опции COBOL в атрибуте ENVIRONMENT для файла ПЛ/1. Опция предписывает планировать структуру согласно правилам Кобола.

Таблица 5.6

Соответствие данных ПЛ/1 и Кобола

Кобол				ПЛ/1			
Тип	Длина, байт	Выравнивание: да	Выравнивание: нет	Тип	Длина, байт	Выравнивание: да	Выравнивание: нет
COMPUTATIONAL 1 —4	2	Полуслово	Байт	REAL FIXED BIN	2	Полуслово	Байт
5 —9	4	Слово	»	» » »	4	Слово	»
10—18	8	»	»	—	—	—	—
COMPUTATIONAL-1	4	»	»	REAL FLOAT DEC	4	Слово	Байт
COMPUTATIONAL-2	8	Двойное слово	»	» » »	8	Двойное слово	»
COMPUTATIONAL-3	~	Байт	»	REAL FIXED DEC	~	Байт	»
DISPLAY	~	»	»	CHAR	~	»	»

Данные ПЛ/1 и языка Ассемблера хорошо согласуются, если в ПЛ/1 они объявлены ALIGNED (см. табл. 5.7).

Таблица 5.7

Соответствие данных ПЛ/1 и языка Ассемблера

Язык Ассемблера			ПЛ/1			
Тип	Длина, байт	Выравнивание	Тип	Длина, байт	Выравнивание: да	Выравнивание: нет
H	2	Полуслово	REAL FIXED BIN	2	Полуслово	Байт
F	4	Слово	» » »	4	Слово	»
E	4	»	REAL FLOAT BIN	4	»	»
E	4	»	REAL FLOAT DEC	4	»	»
D	8	Двойное слово	REAL FLOAT BIN	8	Двойное слово	»
D	8	»	REAL FLOAT DEC	8	»	»
D	~	Байт	REAL FIXED DEC	~	Байт	»
B	~	Бит	BIT	~	»	Бит
C	~	Байт	CHAR	~	»	Байт
Z	~	»	PICTURE	~	»	»

Передача данных между модулями, полученными трансляцией с разных языков, может быть организована либо посредством списка параметров, либо при помощи общих областей памяти.

Пример 4. Передача массива из модуля ПЛ/1 в модуль Фортрана.

Модуль ПЛ/1

```
PRIM4:  PROCEDURE OPTIONS(MAIN);
        DCL TAB(6,3,2) FIXED BIN(31),
            ABP FIXED DEC(1,0) BASED(B);
            B=ADDR(TAB);
            .
            .
            .
            CALL FORMOD(ABP);
            . . . . . .
END PRIM4;
```

Модуль Фортрана

```
SUBROUTINE FORMOD(TAB)
INTEGER TAB (2,3,6)
      . . . . . .
END
```

Объявлена арифметическая базированная переменная ABP. При помощи указателя B производится совмещение переменной ABP с началом массива TAB в памяти машины. Идентификатор ABP указывается вместо идентификатора массива TAB при обращении к подпрограмме FORMOD. В списке параметров, таким образом, появится не адрес информационного вектора массива, а адрес начала массива. В результате модуль Фортрана получает доступ к элементам массива модуля ПЛ/1.

Пример 5. Передача массива из модуля Фортрана в модуль ПЛ/1.

Модуль Фортрана

```
INTEGER TAB(2,3,6)
.
.
CALL PRIM5(TAB)
      . . .
END
```

Модуль ПЛ/1

```
PRIM5: PROCEDURE(VP) OPTIONS(MAIN);
DCL TAB(6,3,2) FIXED BIN(31) BASED(B),
      VP FIXED DEC(1,0);
    B=ADDR(VP);
      .  .
END PRIM5;
```

Модуль Фортрана вызывает модуль PRIM5 ПЛ/1 и передает ему массив TAB. В списке параметров появляется адрес начала этого массива. В модуле ПЛ/1 этот адрес будет иметь формальная вспомогательная переменная VP. Указатель B совмещает в памяти начало базированного массива TAB ПЛ/1 с переменной VP и тем самым с началом массива TAB Фортрана.

В примерах 4 и 5 согласование нумерации элементов массивов ПЛ/1 и Фортрана не предусмотрено: нужно использовать элементы с противоположной индексацией.

Список параметров для скалярных (элементных) данных ПЛ/1 имеет обычный вид (см. рис. 5.12). В других случаях (при передаче массивов, структур, строк) список параметров состоит не из адресов параметров, а из адресов их информационных векторов. Это несоответствие в способах передачи параметров можно устранить использованием в модуле ПЛ/1 арифметических базированных переменных (см. примеры 4 и 5).

Передача данных между модулями посредством поименованных общих областей (COMMON в Фортране и STATIC EXTERNAL в ПЛ/1) возможна при выполнении определенных условий. Прежде всего необходимо учитывать, что в начале общей области ПЛ/1 может располагаться информационный вектор структуры, поэтому в соответствующей общей области Фортрана первым необходимо указать фиктивный (выравнивающий) массив равной длины. Кроме того, данные общей области Фортрана должны быть искусственно выравнены, так как при компиляции их выравнивание не производится.

Связь по управлению модуля ПЛ/1 с модулями других языков также имеет особенности, которые во многом определяются спецификой динамического распределения памяти компилятором ПЛ/1. Динамическое распределение памяти организуют обслуживающие модули (подпрограммы) библиотеки ПЛ/1. Эти подпрограммы образуют так называемую *среду*, необходимую для работы модулей ПЛ/1. Указанием на установление среды является объявление процедуры *главной* посредством описателя OPTIONS (MAIN). Любой модуль ПЛ/1, не имеющий опции MAIN, предполагает среду установленной. Вызывающий модуль ПЛ/1 указывает вызываемому модулю ПЛ/1 адрес динамической области памяти (DSA). Вызываемый модуль извлекает этот адрес из третьего слова области сохранения вызывающего модуля. Область сохранения вызывающего модуля находится в начале DSA этого модуля. Адрес области сохранения вызывающего модуля передается вызываемому в регистре 13 (в соответствии с соглашениями о связях). Вызывающий модуль другого языка высокого уровня не обеспечивает вызываемый модуль ПЛ/1 адресом новой DSA, а без этого любой модуль ПЛ/1, кроме главного, правильно выполнить невозможно. Следовательно, модуль Фортрана или Кобола может вызывать только главный модуль ПЛ/1, который сам устанавливает среду.

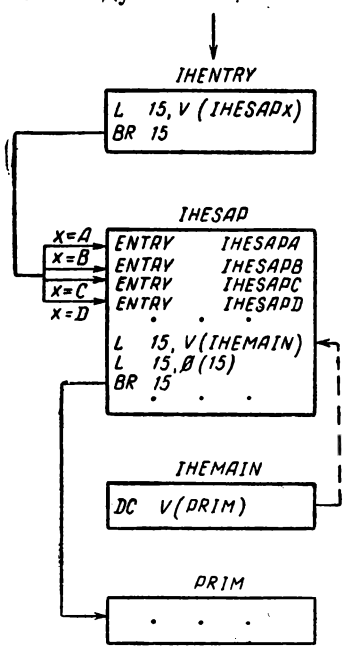


Рис. 5.13. Схема входа в главный модуль ПЛ/1

Схема, представленная на рис. 5.13, реализуется в режимах PCP и MFT. Компилятор ПЛ/1 при обработке главного модуля генерирует две вспомогательные программные секции IHEMAIN и IHENTRY. Первая хранит адрес входа в модуль PRIM, вторая получает управление первой и обеспечивает переход к нужной точке входа подпрограммы IHESAP, устанавливающей среду. Выбор точки входа определяется опцией OPT компилятора и способом передачи параметра модулю. Если параметр передается из поля PARM оператора EXEC, точкой входа будет IHESAPA (при OPT=0) или IHESAPC (при OPT=1 или OPT=2). Если параметры передаются из вызывающего модуля, используется вход IHESAPB (при OPT=0) или IHESAPD (при OPT=1 или OPT=2). Очевидно, начальный вход можно осуществить в нужной точке, минуя IHENTRY. После установления среды происходит переход к модулю PRIM. При этом адрес точки входа выбирается из секции IHEMAIN.

При вызове из модуля ПЛ/1 модуля другого языка проблема среды ПЛ/1 не возникает.

Таким образом, и в первом случае (вызываемый модуль ПЛ/1), и во втором случае (вызывающий модуль ПЛ/1) связь по управлению возможна. Необходимо, однако, следить за тем, чтобы при выполнении программы не создавались неблагоприятные ситуации, связанные с использованием свободной области памяти различными модулями. Кроме того, различия в организации передачи значения функции ПЛ/1 и Фортрана (в Коболе аппарат функций отсутствует) не позволяют использовать функции одного языка в модуле другого языка. Речь идет о всех функциях, вызываемых посредством указателя функции.

Составление задания на компиляцию, редактирование и исполнение программы, состоящей из модулей различных языков, имеет ряд особенностей. Некоторые из них иллюстрируются приведенными ниже примерами 6 и 7. При составлении задания необходимо, в частности, учитывать следующее:

1. При вызове главного модуля ПЛ/1 из модуля другого языка должно быть указано не имя входа в модуль ПЛ/1, а имя входа в обслуживающую подпрограмму IHESAP (обычно указывают имя IHESAPD). Поскольку это имя состоит из семи символов, а в Фортране допускается не более шести, в операторе CALL Фортрана нужно записать фиктивное имя. На этапе редактирования оно заменяется при помощи управляющего предложения CHANGE редактора связей на нужное имя (это показано в примере 6). Предложение CHANGE должно предшествовать в наборе данных основного ввода редактора предложению INCLUDE, включающему объектные модули. При этом модуль Фортрана должен располагаться в наборе данных первым, так как предложение CHANGE относится только к первому объектному модулю. Это означает, что при использовании каталогизированных процедур модуль Фортрана должен компилироваться первым.

2. Чтобы обеспечить автоматическое разрешение внешних ссылок в модулях различных языков, необходимо предусмотреть библиотеки соответствующих компиляторов для пункта редактирования. В примерах 6 и 7 библиотеки ПЛ/1 и Фортрана сцеплены в одном наборе данных SYSLIB. К библиотеке SYS1.FORTLIB, указанной в каталогизированной процедуре, добавляется библиотека SYS1.PL1LIB.

3. В задании следует указать точку входа в загрузочный модуль. В примере 6 первым должен выполняться модуль Фортрана, поэтому в предложении ENTRY указано служебное имя MAIN главной программы Фортрана. В примере 7 осуществляется начальный вход во вспомогательную программную секцию IHENTRY.

4. Пункт выполнения в каталогизированной процедуре нужно дополнить, если это необходимо, операторами DD, обеспечивающими выполнение ввода-вывода в модулях различных языков.

5. Если в программе имеется несколько модулей ПЛ/1 с опцией MAIN, редактор связей включает только секцию IHEMAIN, появившуюся первой, и игнорирует остальные. Это означает, что в программе может быть только один модуль ПЛ/1 с опцией MAIN.

6. Объектный модуль ПЛ/1, содержащий общую секцию, должен предшествовать модулю Фортрана с такой же общей секцией на входе редактора. В противном случае информационный вектор структуры ПЛ/1 в общей секции заполнен не будет (редактор связей обрабатывает только первое из нескольких одинаковых имен).

Пример 6. Задание на компиляцию, редактирование и исполнение. (Программа состоит из вызывающего модуля Фортрана и вызываемого модуля ПЛ/1.)

```
//FORTPL   JOB
//FOR      EXEC FORTGC
//FORT.SYSIN DD  *
      DO 2 I=1,5
      READ(5,1) X
      CALL FICT(X,Y)
   2  WRITE(6,8) X,Y
      STOP
   1  FORMAT(E9.2)
   8  FORMAT(E9.2,E2Ø.3)
      END
/*
//PL   EXEC PL1LFC
//PL1L.SYSIN DD  *
 PODPR: PROCEDURE(X1,Y1)  OPTIONS(MAIN);
        DCL(X1,Y1)  FLOAT  DEC(6);
        Y1=1.+.5*X1+1.5*X1*X1;
  END PODPR;
/*
//ENDPROC EXEC FORTGLG
//LKED.SYSLIB  DD  DSN=SYS1.PL1LIB,DISP=SHR
//               DD  DSN=SYS1.FORTLIB,DISP=SHR
//LKED.OBJMOD DD  DSN=&LOADSET,DISP=(OLD,DELETE)
//LKED.SYSIN   DD  *
 CHANGE FICT(IHESAPD)
```

```
 INCLUDE OBJMOD
 ENTRY MAIN
/*
//GO.SYSIN DD  *
    | Исходные данные |
/*
//
```

Пример 7. Задание на компиляцию, редактирование и исполнение. (Программа состоит из вызывающего модуля ПЛ/1 и вызываемого модуля Фортрана.)

```
//PLFORT JOB
//PL       EXEC PL1FC
//PL1L.SYSIN DD *
  S:  PROCEDURE OPTIONS(MAIN);
      DCL(X,Y) FLOAT DEC(6);
      DO I=1 TO 5;
      GET EDIT (X) (E(9,2));
      CALL AM(X,Y);
      PUT EDIT(X) (E(9,2)) (Y) (E(20,3));
      END;
  END S;
/*
//FOR EXEC FORTGCLG
//FORT.SYSIN DD  *
        SUBROUTINE AM(X1,Y1)
        Y1=1.+X1*(1.5+X1*5)
        RETURN
        END
/*
//LKED.SYSLIB  DD  DSN=SYS1.PL1LIB,DISP=SHR
//               DD  DSN=SYS1.FORTLIB,DISP=SHR
//LKED.SYSIN   DD   *
  ENTRY IHENTRY
/*
//GO.FT05F001   DD  DDNAME=S
//GO.SYSPRINT  DD  SYSOUT=A
//GO.SYSIN      DD   *
    | Исходные данные |
/*
//
```

5.4. УТИЛИТЫ

5.4.1. Общие сведения

Утилитами называют вспомогательные сервисные программы системы программирования ОС ЕС, обеспечивающие работу с наборами данных. Чтобы выполнить утилиту, необходимо вызвать ее при помощи оператора EXEC языка управления заданиями. Утилиты можно также вызывать как подпрограммы. В этом случае нужно соблюдать стандартные соглашения о связях.

Каждая утилита предоставляет пользователю ряд сервисных возможностей. Требуемые функции утилиты можно указать по-

средством управляющих операторов этой утилиты. Некоторые функции являются общими для группы утилит.

Выполнение утилиты должно быть обеспечено необходимыми наборами данных. Их описывают в операторах DD языка управления заданиями для пункта выполнения утилиты. Обязательными наборами данных для всех утилит являются управляющий набор данных и набор данных для сообщений.

Управляющий набор данных (dd-имя SYSIN) содержит управляющие операторы утилиты и обычно располагается во входном потоке. Набор данных для сообщений (dd-имя SYSPRINT) служит для вывода информации об используемых операторах утилиты и об ошибках. Многие утилиты требуют также указания входного, выходного и рабочих наборов данных.

Операторы управления заданиями пункта выполнения утилиты можно объединить в процедуру и каталогизировать ее. Это упрощает использование утилиты программистами.

При завершении работы утилиты (успешном или по причине ошибок) устанавливается код возврата, который передается вызвавшей программе в регистре 15. Код используется для определения уровня ошибки, возникшей при выполнении утилиты, и может иметь значения 00, 04, 08, 12 и 16. Код 00 указывает на успешное завершение работы. Код 04 означает частичное выполнение утилиты. Другие значения кода возврата устанавливаются обычно при обнаружении серьезных ошибок, не позволяющих продолжать выполнение сервисной программы.

Назначения отдельных утилит и их возможности рассмотрены ниже.

Утилита IEBGENER осуществляет преобразование набора данных и предоставляет следующие возможности:

копирование последовательного набора данных или раздела библиотечного набора данных;

преобразование в библиотечный набор данных последовательного набора данных или раздела библиотечного набора данных;

расширение (пополнение) существующего библиотечного набора данных;

редактирование набора данных.

Копию набора данных можно получить на любом носителе (на магнитных лентах или на дисках, на бумажной ленте или перфокартах).

В результате редактирования данные представляются в нужном порядке и формате. Можно, например, преобразовать данные в наборе из упакованного в распакованный десятичный формат.

Возможности утилиты используются посредством управляющих операторов утилиты GENERATE (производить), EXITS (выходы), LABELS (метки), MEMBER (раздел, член), RECORD (запись).

Утилита IEBPTPCH обеспечивает печать и перфорацию наборов данных. Ее возможности таковы:

печать (перфорация) последовательного или библиотечного набора данных;

печать (перфорация) разделов библиотечного набора данных;
печать (перфорация) записей из последовательного или библиотечного набора данных;
печать (перфорация) оглавления библиотечного набора данных;
печать (перфорация) с редактированием библиотечного или последовательного набора данных.

Управление утилитой осуществляется операторами утилиты PRINT (печать), PUNCH (перфорация), TITLE (заголовок), EXITS, MEMBER, RECORD, LABELS.

Формат выводимых данных описывают при помощи операндов управляющих операторов утилиты. Можно, например, указать количество символов в строке печати и требуемый тип данных (шестнадцатеричные, буквенно-цифровые, упакованные или распакованные десятичные). Если формат не указан, данные выводятся в стандартном формате утилиты. В частности, данные печатаются без преобразования типа, а каждая строка содержит 12 групп по восемь знаков. Одна группа отделяется от другой двумя пробелами.

Утилита IEBUPDTE предназначена для обслуживания символических библиотек. Функции утилиты:

добавление, копирование, замена разделов или наборов данных;

добавление, вычеркивание, замена, перенумерация записи внутри раздела или набора данных;

присваивание порядкового номера записи внутри раздела или набора данных;

преобразование последовательного набора данных в библиотечный, и наоборот.

Утилиту, в частности, можно использовать для создания символических библиотек. С ее помощью можно добавлять разделы в существующую библиотеку (например, добавить каталогизированную процедуру в библиотеку процедур).

Управляющие операторы утилиты: ADD (добавить), REPL (заменить, переместить), CHANGE (изменить), REPRO (копировать), NUMBER (перенумеровать), DELETE (вычеркнуть, уничтожить), LABEL (метка), ALIAS (псевдоним), ENDUP (конец данных).

Утилита IEBCOPY работает только с библиотечными наборами данных и обеспечивает:

копирование одного или нескольких наборов данных;
объединение наборов данных;

копирование или исключение из копирования отдельных разделов набора данных.

Программист указывает нужные функции утилиты посредством управляющих операторов утилиты COPY (копировать), SELECT (выбрать), EXCLUDE (исключить).

Утилита IEBCOMPR позволяет сравнить два одинаково организованных набора данных. Можно, например, проверить правильность копии последовательного или библиотечного набора

данных. Управляющие операторы утилиты: COMPARE (сравнить), EXITS, LABELS.

Утилита IEBDG служит для создания набора тестовых данных, которые могут использоваться как вспомогательное средство отладки программ.

Утилита IEBEDIT создает выходной поток заданий путем выборочного (по указанию программиста) копирования входных заданий.

Утилита IEBISAM обеспечивает копирование индексно-последовательных наборов данных.

Все рассмотренные выше утилиты могут выдавать диагностические сообщения с префиксом IEB. Утилиты, имеющие префикс сообщений IEH, описаны ниже.

Утилита IEHMOVE копирует наборы данных. Основными функциями утилиты являются пересылка или копирование следующих наборов данных:

набора данных, находящегося на одном или более (до пяти) томах;

группы каталогизированных наборов данных;

каталога или его части;

тома набора данных.

Кроме этого, утилиту можно использовать для объединения разделов нескольких библиотечных наборов, включения, исключения или замещения отдельных разделов, переименования пересылаемых или копируемых разделов.

Пересылка набора данных из томов прямого доступа сопровождается вычеркиванием исходного набора. Если пересылается каталогизированный набор данных, каталог корректируется. При копировании исходные данные сохраняются, и каталог не меняется.

Основные управляющие операторы утилиты: MOVE (переслать), COPY (копировать). Вспомогательные управляющие операторы утилиты: INCLUDE (включить), EXCLUDE (исключить), REPLACE (заменить), SELECT (выбрать).

Другие утилиты предназначены в основном для обслуживания системных управляющих данных и разметки магнитных лент и томов прямого доступа. Так, утилита IEHPROGM обеспечивает создание и обслуживание системных управляющих данных, а утилита IEHLIST печатает эти данные. Утилита IEHIOSUP дает возможность модифицировать библиотеки супервизора. Утилита

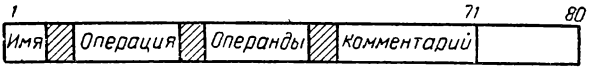


Рис. 5.14. Формат управляющих операторов утилит

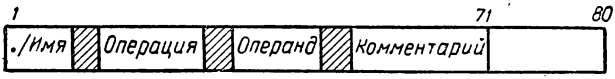


Рис. 5.15. Формат управляющих операторов утилиты IEBUPDTE

IEHDASDR служит для разметки томов прямого доступа, а утилита IEHINITT — для разметки магнитных лент.

Формат управляющих операторов всех утилит (рис. 5.14), за исключением утилиты IEBUPDTE (рис. 5.15), одинаков. Имя и комментарий обязательны не для всех операторов, некоторые из них не имеют операндов. Можно продолжить оператор на следующих картах, начиная с колонки 4—16. Признак продолжения — любой символ, кроме пробела, в колонке 72 карты с прерванным оператором.

5.4.2. Примеры применения

Пример 1. Применение утилиты IEBPTPCH для печати последовательного набора данных.

```
//UTILITA1 JOB
//STEP1  EXEC  ASMFC
//ASM.SYSGO DD DSN=&LOADSET,UNIT=SYSSQ,SPACE=(80,(100,50)),
//          DISP= (MOD,PASS),DCB=(BLKSIZE=400)
//ASM.SYSIN  DD  *
```

Исходный модуль Ассемблера (см. рис. 4.9)

```
/*
//STEP2 EXEC ASMFC
//ASM.SYSGO DD DSN=&LOADSET,DISP=(MOD,PASS)
//ASM.SYSIN  DD *
```

Исходный модуль Ассемблера (см. рис. 4.10)

```
/*
//PECHOM    EXEC  PGM=IEBPTPCH
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    DSN=&LOADSET,DISP=(OLD,DELETE)
//SYSUT2    DD    SYSOUT=A
//SYSIN     DD    *
  PRINT     TOTCONV=XE
/*
//
```

Задание UTILITA1 состоит из трех пунктов. Пункт STEP1 обеспечивает ассемблирование исходного модуля, текст которого показан на рис. 4.9. Полученный объектный модуль записывается во временный последовательный набор данных LOADSET, описанный в операторе DD с dd-именем SYSGO. В пункте STEP2 ассемблируется исходный модуль, представленный на рис. 4.10. Выработанный в этом пункте объектный модуль добавляется в набор данных LOADSET.

Последний пункт PECHOM предписывает напечатать объектные модули, созданные в предыдущих пунктах задания. Для этого используется утилита IEBPTPCH. В качестве входного набора данных (dd-имя SYSUT1) указан последовательный набор данных LOADSET, содержащий объектные модули. Выходным набором данных (dd-имя SYSUT2) является устройство печати. Для вывода сообщений утилиты предусмотрено устройство печати (dd-имя набора данных — SYSPRINT). Управляющий набор данных (dd-имя SYSIN) находится во вводном потоке и содержит один управляющий оператор PRINT. Этот оператор предписывает вывести (печатать) данные из входного набора в выходной. Операнд TOTCONV=XE указывает, что объектные модули должны печататься в шестнадцатеричном представлении. Отсутствие других управляющих

операторов утилиты означает, что данные будут напечатаны в стандартном формате утилиты.

Пример 2. Применение утилиты IEBPTPCH для печати раздела библиотечного набора данных.

```
//UTILITA2  JOB
//STEP1     EXEC  ASMFC
//ASM.SYSGO  DD  DSN=&LOADSET,UNIT=SYSSQ,SPACE=(80,(100,50)),
//               DISP=(MOD,PASS),DCB=(BLKSIZE= 400)
//ASM.SYSIN DD  *
```

Исходный модуль Ассемблера (см. рис. 4.9)

```
/*
//STEP2  EXEC  ASMFCL
//ASM.SYSGO DD DCB=(BLKSIZE=400),DISP=(MOD,PASS)
//ASM.SYSIN  DD *
```

Исходный модуль Ассемблера (см. рис. 4.10)

```
/*
//LKED.SYSLIN       DD     DCB=(BLKSIZE=400)
//PECHZM            EXEC   PGM=IEBPTPCH
//SYSPRINT          DD     SYSOUT=A
//SYSUT1            DD     DSN=&GOSET(GO),DISP=(OLD,DELETE)
//SYSUT2            DD     SYSOUT=A
//SYSIN             DD     *
 PRINT TYPORG=PO,TOTCONV=XE,MAXNAME=1
 MEMBER NAME=GO
/*
//
```

Обрабатываются те же исходные модули, что и в примере 1. Задание UTILITA2 состоит из трех пунктов. Пункт STEP1 не отличается от пункта с таким же наименованием в примере 1. В пункте STEP2 предусмотрено ассемблирование второго исходного модуля и совместное редактирование двух объектных модулей. Полученный загрузочный модуль помещается в раздел GO библиотечного набора данных GOSET, описанного в операторе DD с dd-именем SYSLMOD каталогизированной процедуры ASMFCL.

Пункт PECHZM задания предписывает напечатать загрузочный модуль. Назначение операторов DD этого пункта аналогично назначению соответствующих операторов пункта PECHOM в примере 1. Управляющий оператор PRINT утилиты показывает, что печатаются данные в шестнадцатеричном представлении (операнд TOTCONV=XE) из набора данных, организованного по разделам (операнд TYPORG=PO). Операнд MAXNAME=1 этого оператора указывает, что количество управляющих операторов MEMBER утилиты, находящихся в управляющем наборе данных не превышает 1. Операнд NAME=GO оператора MEMBER указывает имя раздела, из которого печатаются данные (загрузочный модуль).

ПРИЛОЖЕНИЕ 1

УПРАВЛЯЮЩИЕ ОПЕРАТОРЫ ЯЗЫКА УПРАВЛЕНИЯ ЗАДАНИЯМИ

Оператор JOB

Позиционные параметры:

Учетная информация — см. п. 2.2.1.
Имя программиста — см. п. 2.2.1.

Ключевые параметры:

CLASS=класс задания (только MFT и MVT)

«Класс задания» — латинская буква от A до O, определяющая класс задания. Стандартное значение CLASS=A.

COND=((код,оператор),...,(код,оператор))

Использование параметра описано в п. 2.2.1.

MSGCLASS=выходной класс (только MFT и MVT)

«Выходной класс» — буква от A до Z или цифра от 0 до 9, определяющая класс, в который направляются сообщения планировщика заданий. Стандартное значение MSGCLASS=A.

MSGLEVEL=(операторы,сообщения)

Использование параметра описано в п. 2.2.1. Стандартное значение MSGLEVEL=(Ø,1).

PRTY=приоритет (только MFT и MVT)

«Приоритет» — число от 0 до 13 определяющее приоритет задания. Стандартное значение PRTY=1.

RD=запрос

«Запрос»: R — разрешить автоматическое повторение пункта задания при его аварийном завершении (или сбое системы в MFT и MVT), NC — подавить действие макрокоманды CHKRT и запретить автоматическое повторение пункта задания, NR — запросить разрешение макрокоманды CHKRT для установления контрольной точки и запретить автоматическое повторение пункта задания, RNC — разрешить повторение пункта задания и подавить действие макрокоманды CHKRT.

REGION=NK (только MVT)

N — четное десятичное число, не превосходящее 16383 и определяющее размер области основной памяти в килобайтах, требующейся для выполнения задания. Стандартное значение REGION=5ØK.

RESTART=({ * / имя пункта / имя пункта.имя пункта процедуры } ,имя контрольной точки)

Параметр используется при повторном вводе задания, закончившегося аварийно. Первый подпараметр (*) — означает, что нужно повторить задание с первого пункта, «имя пункта» и «имя пункта.имя пункта процедуры» — повторить задание с указанного пункта; второй подпараметр («имя контрольной точки») — повторить задание с данной точки внутри пункта (для этого во время первоначального выполнения должна быть выполнена макрокоманда CHKRT). Второй подпараметр не обязателен.

TIME=(минуты,секунды) (только MFT и MVT)

Использование параметра описано в п. 2.2.1. Стандартное значение TIME=3Ø.

TYPRUN=HOLD (только MFT и MVT)

Параметр предписывает задержать выполнение задания до тех пор, пока оператор не введет через пультовую машинку команду RELEASE.

ROLL=(x,y) (только MVT)

Подпараметр x имеет два значения: YES — пункты задания могут быть свернуты, NO — пункты задания не могут быть свернуты. Подпараметр *y* также имеет два значения: YES — пункты задания могут свертывать другие пункты заданий, NO — пункты задания не могут свертывать другие пункты заданий. Оба подпараметра обязательны. Стандартное значение ROLL=(YES,NO).

Оператор EXEC

Позиционный параметр:

{ PGM= { имя программы / *.имя пункта.dd-имя / *.имя пункта.имя пункта процедуры.dd-имя } }
{ { PROC=имя процедуры / имя процедуры } }

Ключевые параметры:

Для модификации ключевых параметров в процедурах пишут:
ключевое слово.имя пункта процедуры=значение

ACCT=(учетная информация)

«Учетная информация» — список подпараметров, разделенных запятыми (длина списка до 142 знаков, включая запятые). Подпараметр, содержащий специальные знаки, заключают в апострофы. Параметр ACCT используется, если в систему включена программа обработки учетной информации.

COND=({ (код,оператор) / (код,оператор, имя пункта) / (код,оператор,имя пункта процедуры) } ,..., { EVEN / ONLY })

Использование параметра описано в п. 2.2.2.

DPRTY=(x,y) (только MVT и MFT с подзадачами)

Подпараметры x и y — числа от 0 до 15, используемые системой для определения диспетчерского приоритета задачи по формуле

«Диспетчерский приоритет»=x×16+y

PARM=значение

Использование параметра описано в п. 2.2.2.

RD=запрос

См. параметр RD оператора JOB.

REGION=значение (только MVT)

См. параметр REGION оператора JOB. В операторе EXEC параметр REGION относится только к пункту задания и используется, если не определен параметр REGION в операторе JOB. Стандартное значение REGION=5ØK.

ROLL=значение (только MVT)

См. параметр ROLL в операторе JOB. В операторе EXEC параметр ROLL относится только к пункту задания и используется, если не определен параметр ROLL в операторе JOB.

TIME=(минуты,секунды)

Использование параметра описано в п. 2.2.2. Стандартное значение TIME=3Ø.

Оператор DD

Позиционный параметр:

{ * / DATA / DUMMY }

Использование параметра описано в п. 2.3.3.

Ключевые параметры:

При описании ключевых параметров «ссылка на dd-имя» обозначает одно из следующих трех значений:

*.dd-имя
*.имя пункта.dd-имя
*.имя пункта.имя пункта процедуры.dd-имя

AFF=dd-имя

Использование параметра описано в п. 2.3.12.

DCB= { (список подпараметров) / ({ ds-имя / ссылка на dd-имя } ,список подпараметров) }

Использование параметра описано в п. 2.3.11.

DDNAME=dd-имя

Параметр указывает, что набор данных определен в том же пункте задания позже оператором DD с заданным dd-именем.

DISP=(состояние,диспозиция,условная диспозиция)

Использование параметра описано в п. 2.3.6. Стандартное значение DISP=NEW.

$$\left\{\begin{matrix}\text{DSNAME}\\\text{DSN}\end{matrix}\right\}=\left\{\begin{matrix}\text{имя набора данных}\\\text{\&имя набора данных}\\\text{\&\&имя набора данных}\\\text{ссылка на dd-имя}\end{matrix}\right\}$$

Использование параметра описано в п. 2.3.5. Для индексно-последовательного набора данных «имя набора данных» пишут в форме: «имя(имя области)», где «имя области» — INDEX или PRIME, или OVFLOW; для поколения данных «имя набора данных» — «имя(номер поколения)», где «номер поколения» — нуль или целое со знаком.

LABEL=(номер,тип меток,вид защиты,вид обработки,срок хранения)

Использование параметра описано в п. 2.3.10. Подпараметр «срок хранения» имеет два значения: EXPDT=ГГДДД, где ГГ — год (например, 78 означает 1978 год), ДДД — порядковый номер дня в году, после которого набор данных можно уничтожить (ØØ1 означает 1 января); RETPD=ДДД — число дней хранения.

SEP=(список dd-имен)

Использование параметра описано в п. 2.3.12.

$$\text{SPACE}=(\text{единицы},(\text{количество,приращение},\left\{\begin{matrix}\text{оглавление}\\\text{индекс}\end{matrix}\right\}),\text{RLSE},\left\{\begin{matrix}\text{CONTIG}\\\text{MXIG}\\\text{ALX}\end{matrix}\right\},\text{ROUND})$$

Использование параметра описано в п. 2.3.8 и 2.3.12. «Индекс» — размер индекса в цилиндрах для индексно-последовательного набора данных. При указании индекса «приращение» нужно опустить, а в параметре DCB нужно записать подпараметр DSORG=IS или DSORG=ISU.

$$\text{SPACE}=(\text{ABSTR},(\text{количество,адрес})\left\{\begin{matrix}\text{,оглавление}\\\text{,индекс}\end{matrix}\right\})$$

Размещение набора данных с определенной дорожки. «Количество» — количество дорожек, «адрес» — относительный адрес первой дорожки, занимаемой набором данных (дорожку Ø использовать нельзя). «Оглавление» имеет обычное значение для библиотек (см. п. 2.3.8), а «индекс» — число дорожек для индекса, кратное числу дорожек в цилиндре (для индексно-последовательного набора данных «количество» должно быть кратно числу дорожек в цилиндре, а «адрес» — первая дорожка цилиндра, но не первого цилиндра в томе).

$$\text{SPLIT}=\left\{\begin{matrix}\text{(n,CYL,(величина,приращение))}\\\text{(\%,размер блока,(величина,приращение))}\end{matrix}\right\}$$

Использование параметра описано в п. 2.3.12.

$$\text{SUBALLOC}=(\text{единицы},(\text{количество,приращение,оглавление}),\left\{\begin{matrix}\text{dd-имя,}\\\text{имя пункта.dd-имя}\\\text{имя пункта.имя пункта процедуры.dd-имя}\end{matrix}\right\})$$

Использование параметра описано в п. 2.3.12.

SYSOUT=(имя класса,имя программы,номер формата),OUTLIM=число

Использование параметра описано в п. 2.3.3. Все подпараметры, кроме первого, не обязательны. В режиме PCP используется только первый подпараметр. «Имя класса» — буква от A до Z или цифра от 0 до 9. Буква или цифра определяет класс выходного потока. «Имя программы» указывает специальную программу вывода, находящуюся в общей системной библиотеке. «Номер формата» — комбинация до четырех букв или цифр (указание специального формата вывода).

«OUTLIM=число» — указание предельного числа записей в выходном наборе данных (применяется только при включении в систему средств сбора статистики).

$$\text{UNIT}=\left\{\begin{array}{l}(\text{устройство},\left\{\begin{array}{l}\text{n}\\\text{P}\end{array}\right\},\ \text{DEFER,SEP}=(\text{список dd-имен}))\\\text{AFF}=\text{dd-имя}\end{array}\right\}$$

Использование параметра описано в п. 2.3.7 и 2.3.12. Подпараметр n — число устройств. Если набор данных каталогизирован и занимает несколько томов, то буква P вместо n предписывает назначить каждому тому отдельное устройство (тома могут устанавливаться параллельно). DEFER предписывает отложить установку тома до тех пор, пока набор данных не потребуется фактически для выполнения операций ввода-вывода.

$$\text{VOLUME}=(\text{PRIVATE,RETAIN,номер тома,число томов,}\ \text{SER}=\left\{\begin{array}{l}\text{ds-список серийных номеров})\\\text{ds-имя}\\\text{ссылка на dd-имя}\end{array}\right\})$$

Использование параметра описано в п. 2.3.9.

ПРИЛОЖЕНИЕ 2

ОПЦИИ ТРАНСЛЯТОРОВ И РЕДАКТОРА СВЯЗЕЙ

Значения опций, принимаемые по умолчанию, набраны полужирным шрифтом. Для трансляторов пояснения относятся к утвердительной форме. В скобках приведены допустимые сокращения. Знак «/» разделяет значения опций.

Ассемблер

LOAD/**NOLOAD** — передать объектный модуль редактору.
DECK/NODECK — перфорировать объектный модуль.
LIST/NOLIST — печатать листинг.
TEST/**NOTEST** — подготовить объектный модуль для Тестрана.
XREF/NOXREF — печатать таблицу перекрестных ссылок.
RENT/**NORENT** — проверить реентерабельность объектной программы.
LINECNT=nn — число строк на странице листинга (стандартно 55).
ALGN/NOALGN — печатать сообщения об ошибках выравнивания.
OS/DOS — режим Ассемблера ОС(OS) или режим Ассемблера ДОС(DOS).

Алгол-60

PROGRAM/PROCEDURE(**PG**/PC) — основная программа/процедура.
SHORT/LONG (**SP**/LP) — размещать вещественные переменные в словах основной памяти (SHORT) или в двойных словах (LONG).
DECK/**NODECK** (D/**ND**) — перфорировать объектный модуль.
LOAD/NOLOAD (**L**/NL) — передать объектный модуль редактору.
SOURCE/NOSOURCE(**S**/NS) — печатать исходную программу.
TEST/NOTEST(**T**/NT) — контролировать в объектной программе границы индексов, соответствие числа фактических и формальных параметров, фиксировать трассу.
SIZE=yyyyyy или SIZE=yyyK — число байт основной памяти, доступной компилятору (от 44 до 999K). Стандартное значение SIZE=44K,

SIZE=yyyyyyy или SIZE=yyyyK — число байт основной памяти, доступной компилятору (от 80 до 9999K). Стандартное значение SIZE=80K.

BUF=yyyyyy или BUF=yyyK — память для буферов (входит в SIZE).

SOURCE/NOSOURCE — печатать исходную программу.

CLIST/**NOCLIST** — печатать сокращенный листинг объектной программы.

DMAP/**NODMAP** — печатать имена переменных и их адреса.

PMAP/**NOPMAP** — печатать полный листинг объектной программы.

LIB/NOLIB — в исходной программе есть утверждения BASIS или COPY.

VERB/NOVERB — печатать в листинге объектной программы имена процедур и глаголов.

LOAD/NOLOAD — передать объектный модуль в набор данных редактора связей.

DECK/**NODECK** — перфорировать объектный модуль.

SEQ/NOSEQ — контролировать правильность последовательности утверждений исходной программы.

LINECNT=nn — число строк на странице листинга (стандартно 60).

FLAGW/FLAGE — печатать все сообщения компилятора (FLAGW), печатать все сообщения, кроме предупреждающих (FLAGE).

SUPMAP/**NOSUPMAP** — печатать объектную программу даже при ошибках уровня E.

SPACE1/SPACE2/SPACE3 — разделять слова в листинге одним, двумя или тремя пробелами.

TRUNC/**NOTRUNC** — усекать арифметические значения в соответствии с заданным шаблоном (TRUNC) или в соответствии с размером поля.

QUOTE/**APOST** — выделять литералы кавычками (QUOTE) или апострофами.

XREF/**NOXREF** — печатать таблицу перекрестных ссылок.

ZWB/NOZWB — при сравнении десятичного поля с буквенно-цифровым не учитывать (или учитывать) знак числа.

Фортран стандартный и Фортран оптимизирующий

LIST/**NOLIST** — печатать объектный листинг в ассемблерной форме.

NAME=xxxxxx — имя главной программы (по умолчанию — MAIN).

LINECNT=nn — число строк на странице листинга (стандартно 50).

SOURCE/NOSOURCE — печатать исходную программу.

DECK/**NODECK** — перфорировать объектный модуль.

MAP/**NOMAP** — печатать имена переменных с их адресами.

LOAD/NOLOAD — поместить объектный модуль в набор данных редактора связей.

KOI/**DKOI** — вид кода исходной программы.

ID/**NOID** — включить в объектный модуль номера исходных утверждений, которые вызывают подпрограммы или содержат ссылки к внешним функциям. Эти номера выдаются в листинге, когда печатается трасса. Для записи каждого номера расходуется четыре байта.

Только Фортран оптимизирующий

OPT=Ø — оптимизация не выполняется.

OPT=1 — локальная оптимизация распределения регистров и ветвлений.

OPT=2 — глобальная оптимизация распределения регистров, ветвлений, «чистка циклов», исключение общих выражений.

SIZE=nnnnK — число байт основной памяти, доступной компилятору (от 115 до 9999K). В REGION нужно указать по крайней мере на 10K больше, чем в SIZE.

EDIT/**NOEDIT** — печатать структурный листинг исходной программы, в котором выделены циклы и линейные участки. Для опции EDIT нужно указать OPT=2 и добавить оператор DD вида (в типовых случаях):

//SYSUT1 DD DSN=&UT1,UNIT=SYSSQ,SPACE=(TRK,(4Ø))

XREF/**NOXREF** — печатать таблицу перекрестных ссылок. Для опции XREF нужно добавить оператор DD вида (в типовых случаях):

//SYSUT2 DD DSN=&UT2,UNIT=SYSSQ,SPACE=(TRK,(8)).

ПЛ/1.

SIZE=yyyyyy или SIZE=yyyK — число байт основной памяти, доступной компилятору (от 44 до 999K). Раздел в MFT или область в MVT должны быть по крайней мере на 8K больше. По умолчанию SIZE=999999.

OPT=Ø (O=Ø) — создать минимальную по длине объектную программу.

OPT=1 (O=1) — создать быстродействующую программу.

OPT=2 (O=2) — создать наиболее быстродействующую программу.

STMT/**NOSTMT** (ST/**NST**) — зафиксировать в объектной программе номера исходных утверждений для печати их в диагностических сообщениях.

OBJNM=aaaaaaaa (N=aaaaaaaa) — имя загрузочного модуля (имя начинается с буквы и содержит буквы и цифры). См. п. 3.2.2, пример 4.

EXTDIC/**NOEXTDIC** (ED/**NED**) — увеличить словарь транслятора (для трансляции очень больших программ).

SYNCHKE (SKE) — закончить компиляцию, если при синтаксическом контроле обнаружены ошибки (синтаксический контроль идет до конца).

SYNCHKS (SKS) — закончить компиляцию, если обнаружены серьезные ошибки (синтаксический контроль идет до конца).

SYNCHKT (SKT) — закончить компиляцию, если встретились ошибки, препятствующие ее продолжению.

MACRO/**NOMACRO** (M/**NM**) — применить препроцессор.

COMP/NOCOMP (**C**/NC) — после препроцессора выполнить компиляцию.

MACDCK/**NOMACDCK** (MD/**NMD**) — результат работы препроцессора выдать на перфокарты.

CHAR6Ø/CHAR48 (**C**6Ø/C48) — вид литерного набора.

KOI/**DKOI** — вид кода.

SORMGIN=(mmm,nnn,ccc) (SM=(mmm,nnn,ccc)) — формат карт исходного модуля (mmm — номер первого байта, nnn — номер последнего байта, ccc — номер байта, содержащего управляющий символ). Управляющие символы: **B** — перед печатью пропустить одну строку, Ø — две, «—» — три, «+» — запретить пробел перед печатью, 1 — начать новую страницу. Стандартное значение SM=(2,72).

LOAD/NOLOAD (**LD**/NLD) — передать объектный модуль редактору связей.

LINECNT=xxx (LC=xxx) — число строк на странице листинга (**LC**=5Ø).

OPLIST/NOOPLIST (**OL**/NOL) — напечатать список действующих опций.

SOURCE2/NOSOURCE2 (**S2**/NS2) — напечатать исходную программу.

SOURCE/NOSOURSE (**S**/NS) — печатать программу после препроцессора.

NEST/**NONEST** (NT/**NNT**) — печатать в листинге уровни блоков и уровни вложенности DO-групп.

ATR/**NOATR** (A/**NA**) — напечатать идентификаторы с их атрибутами.

XREF/**NOXREF** (X/**NX**) — напечатать таблицу перекрестных ссылок.

DECK/**NODECK** (D/**ND**) — перфорировать объектный модуль.

EXTREF/**NOEXTREF** (E/**NE**) — напечатать словарь внешних символов (ESD).

LIST/**NOLIST** (L/**NL**) — напечатать объектную программу в ассемблерной форме.

FLAGW (**FW**) — печатать все диагностические сообщения.

FLAGE (FE) — не печатать предупреждающих сообщений.

FLAGS (FS) — печатать только сообщения о серьезных ошибках.

РПГ

SEQN/**NOSEQN** — проверить возрастание значений в графе «лист — строка».

LOAD/NOLOAD — передать объектный модуль редактору.

DECK/**NODECK** — перфорировать объектный модуль.

LIST/NOLIST — печатать листинг исходной программы.

Редактор связей

NE — приписать модулю атрибут НЕРЕДАКТИРУЕМЫЙ (в модуле нет ESD).

OL — приписать модулю атрибут ТОЛЬКО ЗАГРУЖАЕМЫЙ. Такой модуль можно ввести в основную память только макрокомандой LOAD.

OVLY — создать модуль оверлейной структуры (атрибут ОВЕРЛЕЙ). Эта опция несовместима с опциями RENT, REUS, REFR, SCTR.

RENT — приписать атрибут РЕЕНТЕРАБЕЛЬНЫЙ (модуль могут использовать одновременно несколько заданий). Эта опция несовместима с REUS.

REUS — приписать атрибут ПОВТОРНОИСПОЛЬЗУЕМЫЙ (модуль не может использоваться другим заданием, пока не будет завершено его выполнение).

REFR — приписать модулю атрибут ОБНОВЛЯЕМЫЙ.

SCTR — приписать модулю атрибут ФОРМАТ ВРАЗБРОС (см. п. 4.2.5).

TEST — приписать модулю атрибут ПРОВЕРКА (подготовленный для работы с Тестраном). Опция несовместима с опциями RENT, REUS, REFR, NE.

LET — приписать модулю атрибут ВЫПОЛНИМЫЙ (см. п. 4.2.5).

NCAL — запретить использование аппарата автовызова (см. п. 4.2.5).

XCAL — приписать модулю оверлейной структуры атрибут ВЫПОЛНИМЫЙ при наличии правильных исключающих ссылок (см. разд. 5.2).

SIZE=(V1, V2) — область памяти, доступная редактору (см. п. 4.2.5).

DCBS — сообщение редактору связей, что размер блока для набора данных с dd-именем SYSLMOD указан в параметре DCB оператора DD.

LIST — напечатать список управляющих предложений редактора связей, использованных при обработке.

MAP — печатать план загрузочного модуля (опция несовместима с NE).

XREF — печатать таблицу перекрестных ссылок, включающую план модуля. Эта опция несовместима с опциями MAP и NE.

ЛИТЕРАТУРА

1. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. М., «Мир», 1975.
2. Джермейн К. Программирование на IBM/360. М., «Мир», 1971.
3. Донован Дж. Системное программирование. М., «Мир», 1975.
4. Единая система ЭВМ. Под общ. ред. А. М. Ларионова. М., «Статистика», 1974.
5. Катцан Г. Операционные системы. М., «Мир», 1976.
6. Лебедев В. Н. Введение в системы программирования. М., «Статистика», 1975.
7. Система математического обеспечения ЕС ЭВМ. Под общ. ред. А. М. Ларионова. М., «Статистика», 1974.
8. Супервизор и управление данными. М., «Сов. радио», 1974.
9. Усов С. А. Некоторые результаты измерения производительности ЭВМ. М., ИПМ АН СССР, 1973.
10. Rambousek Jiri. Analise de la structure et de fonctionnement du compilateur PL/1 niveau F. Problèmes poses par le couplage de programmes écrits en utilisant des langages differents. Thèse doct. ing. Univ. Paris — Sud, 1974.

Предисловие 3

Глава 1. **Системы программирования ЕС ЭВМ** 4

1.1. Общие сведения 4
1.1.1. Классификация систем программирования 4
1.1.2. Основные понятия 5
1.2. Система программирования ОС ЕС 8
1.2.1. Наборы данных 8
1.2.2. Состав и схема функционирования системы программирования 10

Глава 2. **Язык управления заданиями** 14

2.1 Задание 14
2.1.1. Формат управляющих операторов 14
2.1.2. Структура задания 16
2.2. Описание задания 19
2.2.1. Оператор задания 19
2.2.2. Оператор пункта задания 22
2.3. Описание наборов данных 25
2.3.1. Общие сведения об операторе описания данных . . 25
2.3.2. Имя оператора описания данных 27
2.3.3. Наборы данных во входном и выходном потоках. Позиционный параметр и параметр SYSOUT 29
2.3.4. Наборы данных на внешних запоминающих устройствах 31
2.3.5. Параметр DSNAME 31
2.3.6. Параметр DISP 33
2.3.7. Параметр UNIT 35
2.3.8. Параметр SPACE 36
2.3.9. Параметр VOLUME 37
2.3.10. Параметр LABEL 39
2.3.11. Параметр DCB 40
2.3.12. Оптимизация размещения наборов данных 41
2.4. Процедуры 45
2.4.1. Описание процедуры 45
2.4.2. Замещение и добавление параметров 48

Глава 3. **Трансляция** 52

3.1. Трансляторы 52
3.1.1. Характеристики 52
3.1.2. Наборы данных 55
3.1.3. Опции 57
3.2. Каталогизированные процедуры трансляции 59
3.2.1. Состав и особенности 59
3.2.2. Использование процедур 62

Глава 4. **Редактирование, загрузка и выполнение** 66

4.1. Объектный модуль 66
4.1.1. Структура 66
4.1.2. Словарь внешних символов 69
4.1.3. Текст 70
4.1.4. Словарь перемещений 71
4.1.5. Пример 72
4.2. Редактор связей 73
4.2.1. Функции 73
4.2.2. Загрузочный модуль 75

4.2.3. Версии 76
4.2.4. Наборы данных 77
4.2.5. Опции 78
4.2.6. Управляющие предложения 80
4.2.7. Каталогизированные процедуры редактирования . . . 85
4.3. Выполнение загрузочного модуля 89
4.3.1. Вызов 89
4.3.2. Наборы данных 90
4.3.3. Каталогизированные процедуры выполнения 92
4.4. Загрузчик 95
4.4.1 Функции загрузчика 95
4.4.2. Вызов загрузчика и его опции 96
4.4.3. Наборы данных загрузчика и объектной программы 97
4.4.4. Каталогизированные процедуры трансляции, загрузки и выполнения 98

Глава 5. Дополнительные сведения о системе программирования . . . 100

5.1. Средства отладки 100
5.1.1. Общесистемные средства 100
5.1.2. Средства отладки в языках и трансляторах 102
5.2. Оверлейные структуры 105
5.2.1. Планирование 105
5.2.2. Описание 108
5.2.3. Некоторые особенности 111
5.3. Межмодульные связи 114
5.3.1. Соглашения о связях. Область сохранения 114
5.3.2. Связь по управлению 116
5.3.3. Связь по данным 119
5.3.4. Особенности связи модулей, транслированных с разных языков 121
5.4. Утилиты 129
5.4.1. Общие сведения 129
5.4.2. Примеры применения 133

Приложение 1. Управляющие операторы языка управления заданиями . . 134

Приложение 2. Опции трансляторов и редактора связей 138

Литература 142

Владлен Николаевич Лебедев,
Александр Павлович Соколов

ВВЕДЕНИЕ В СИСТЕМУ ПРОГРАММИРОВАНИЯ ОС ЕС

Рецензент *Г. В. Пеледов*
Редактор *Л. А. Табакова*
Мл редактор *В. К. Капинская*
Техн. редактор *Л. Г. Челышева*
Худ. редактор *Т. В. Стихно*
Корректоры *Г. А. Башарина, А. Т. Сидорова*
Обложка художника *Т. Н. Погореловой*

ИБ № 372

Сдано в набор 24/VI 1977 г. Подписано к печати 30/XII 1977 г. Формат бумаги 60×90¹/₁₆.
Бумага № 3. Объем 9,0 печ. л. Уч.-изд. л. 9,91. Усл. п. л. 9,0. Тираж 77 000 экз. А14495.
(Тематич. план 1978 г. № 77). Заказ № 5164. Цена 60 коп.

Издательство «Статистика», Москва, ул. Кирова, 39.

Областная типография управления издательств, полиграфии и книжной торговли Ивановского облисполкома, г. Иваново-8, ул. Типографская, 6.

60 коп.